



**ATENCIÓN:**

- Lea atentamente todo el enunciado antes de comenzar a contestar.
  - No se podrán utilizar libros ni apuntes, ni calculadoras de ningún tipo.
  - Los teléfonos móviles deberán permanecer desconectados durante la prueba (apagados, no silenciados).
  - Solamente se corregirán los ejercicios contestados con bolígrafo. Por favor no utilice lápiz.
  - **La duración del examen es de 90 minutos.**
  - **Presente cada ejercicio en una hoja distinta. Si no hace un ejercicio entregue una hoja con el número del ejercicio.**
- 

NOMBRE:

APELLIDOS:

NIA:

**Ejercicio 3 [2.5 puntos]:** Un determinado procesador tiene un único nivel de memoria caché. La caché de 32 KB es asociativa por conjuntos de 4 vías. El tamaño de línea es de 64 bytes. La caché tiene una política de escritura de post-escritura y una política de remplazo LRU.

En esta máquina se está valorando la ejecución del siguiente código:

```
constexpr long max = 4 * 1024 * 1024; // Constante
double x[max], y[max], z[max]; // 2^22 elementos
// Lectura de valores.
// ...
double hx[max], hy[max], hz[max];
for (int i=0; i<max;++i) {
    hx[i] = 0;
    hy[i] = 0;
    hz[i] = 0;
}
for (int i=0; i<max;++i) {
    hx[i] = hx[i] + x[i];
    hy[i] = hy[i] + y[i];
    hz[i] = hz[i] + z[i];
}
```

El vector x se almacena a partir de la dirección de memoria 0x20000000. El resto de vectores se almacenan de forma consecutiva a este.

Se pide:

- a) Determine cuantos conjuntos tiene la caché en cada vía. [0.5 puntos]



- b) Determine la tasa de aciertos de la memoria caché para accesos a datos. Considere que los accesos a la constante max y la variable i no generan acceso a memoria. Considere la caché inicialmente vacía. [1 punto]
- c) Proponga una modificación de código que aplique la técnica de fusión de arrays. [1 punto]

## SOLUCIÓN:

### Apartado A

Caché L1 ( $32\text{KB} = 2^{15}$  bytes)

Cada vía tiene  $2^{15}/2^2$  bytes =  $2^{13}$  bytes. Como cada línea tiene 64 bytes ( $2^6$  bytes), el número de conjuntos por línea será  $2^{13}/2^6 = 2^7$  conjuntos

### Apartado B

Todos los arrays tienen  $2^{22}$  elementos de tipo double. Como cada valor de tipo double ocupa 8 bytes, cada array ocupa exactamente  $2^{25}$  bytes (32 MB).

La dirección de comienzo del vector x es la dirección  $2^{29}$ , que está alineada a la primera línea de caché. El tamaño del array es múltiplo exacto del tamaño de la caché. Como todos los arrays son consecutivos, esto mismo es aplicable para todos los arrays.

En cada línea de caché caben 8 valores de tipo double.

Cuando se ejecuta el primer bucle en la primera iteración se accede a:

hx[0] → fallo. Se trae al conjunto 0, vía 0 los valores hx[0]...hx[7]

hy[0] → fallo. Se trae al conjunto 0, vía 1 los valores hy[0]...hy[7]

hz[0] → fallo. Se trae al conjunto 0, vía 2 los valores hz[0]...hz[7]

Los accesos para  $i=1$  hasta  $i=7$  producen aciertos.

En total se producen por cada bloque 3 fallos y 21 aciertos.

Este patrón se repite para todos los conjuntos de la caché L1. Como en la caché L1 se tienen  $2^7$  conjuntos y cada vez se acceden a  $2^3$  valores, se habrán procesado  $2^{10}$  valores. Lo que da lugar a  $3 \times 2^7$  fallos y  $3 \times 7 \times 2^7$  aciertos



En la segunda vuelta se vuelve a usar la cuarta vía para los valores  $hx[i]$ , y se reusan las vías 0 y 1 para  $hy[i]$  y  $hz[i]$ .

En total se producen 3 fallos y 21 aciertos. Lo que da lugar a  $3 \times 2^7$  fallos y  $3 \times 7 \times 2^7$  aciertos. En la tercera y sucesivas vueltas a la caché, todas las líneas están ya ocupadas por lo que se repite el patrón.

El número total de barridos de la memoria caché se obtiene dividiendo el número de iteraciones del bucle for ( $2^{22}$ ), entre el número de elementos accedidos en una iteración ( $2^{10}$ ), lo que da lugar a un total de  $2^{12}$  iteraciones.

Número total de fallos:  $3 \times 2^7 \times 2^{12} = 3 \times 2^{19}$ .

Número total de aciertos:  $3 \times 7 \times 2^7 \times 2^{12} = 21 \times 2^{19}$ .

A continuación se accede al segundo bucle. Para entonces todos los valores iniciales de la caché han sido expulsados y la situación es equivalente a una caché vacía.

Para  $i=0$  el patrón de accesos es:  $hx[0]$  (V0, fallo),  $x[0]$  (V1, fallo),  $hx[0]$  (V0, acierto),  $hy[0]$  (V2, fallo),  $y[0]$  (V3, fallo),  $hy[0]$  (V2, acierto),  $hz[0]$  (V1, fallo),  $z[0]$  (V0, fallo),  $hz[0]$  (V1, acierto). En total hay 9 accesos (6 fallos + 3 aciertos). Al final los valores que permanecen en caché son ( $hy[0]$ ,  $y[0]$ ,  $hz[0]$ ,  $z[0]$ ). En las siguientes iteraciones se repetirá este patrón, porque las entradas de la caché se van expulsando justo antes de que se puedan reusar.

Por tanto en este segundo bucle se tienen:

Número total de fallos:  $6 \times 2^{22}$  (todos provocan expulsión de bloque)

Número total de aciertos:  $3 \times 2^{22}$

Considerando el código completo se tiene:

$$\text{Fallos} = 3 \times 2^{19} + 6 \times 2^{22} = 2^{19} (3 + 6 \times 2^3) = 51 \times 2^{19}$$

$$\text{Aciertos} = 3 \times 7 \times 2^{19} + 3 \times 2^{22} = 2^{19} (3 \times 7 + 3 \times 2^3) = 2^{19} (21 + 24) = 45 \times 2^{19}$$

$$\text{Tasa de aciertos} = 45 / 96 = 0,468 \rightarrow 46,8\%$$

### Apartado C

`constexpr long max = 4 * 1024 * 1024; // Constante`



Universidad  
Carlos III de Madrid

Departamento de Informática  
Grado en Ingeniería Informática  
Arquitectura de Computadores

Prueba de evaluación continua  
19 de octubre de 2015



```
struct point {  
    double x, y, z;  
};  
point pos[max]; // 2^22 elemento  
// Lectura de valores.  
// ...  
point h[max];  
for (int i=0; i<max;++i) {  
    h[i].x =0;  
    h[i].x = 0;  
    h[i].x = 0;  
}  
for (int i=0;i<max;++i) {  
    h[i].x = h[i].x + pos[i].x;  
    h[i].y = h[i].y + pos[i].y;  
    h[i].z = h[i].z + pos[i].z;  
}
```



**Ejercicio 3 [2.5 puntos]:** Un determinado procesador tiene un único nivel de memoria caché. La caché de 128 KB es asociativa por conjuntos de 8 vías. El tamaño de línea en todos los niveles es de 64 bytes. Todos los niveles tienen una política de escritura de post-escritura y una política de remplazo LRU.

En esta máquina se está valorando la ejecución del siguiente código:

```
constexpr long max = 8 * 1024; // Constante
double m[max][2*max];
// Lectura de valores.
// ...
for (int j=0; j<2*max; ++j) {
    for (int i=0; i<max; ++i) {
        m[i][j] = 0;
    }
}
```

La matriz m se almacena a partir de la dirección de memoria 0x20000000.

Se pide:

- Determine cuantos conjuntos tiene la caché en cada vía. [0.5 puntos]
- Determine la tasa de aciertos de la memoria caché para accesos a datos.  
Considere que los accesos a la constante max y las variables i y j no generan accesos a memoria. Considere la caché inicialmente vacía. [0.75 punto]
- Proponga una modificación de código que aplique la técnica de intercambio de bucles. [1 punto]
- Determine la tasa de aciertos de la memoria caché para acceso a datos del código modificado. [0.25]

## SOLUCIÓN

### Apartado A

Caché L1 (128KB =  $2^{17}$  bytes)

Cada vía tiene  $2^{17}/2^3$  bytes =  $2^{14}$  bytes. Como cada línea tiene 64 bytes ( $2^6$  bytes), el número de conjuntos por línea será  $2^{14}/2^6=2^8$  conjuntos

### Apartado B



La matriz comienza en la dirección  $2^{29}$  que está alineada al principio de la caché. La matriz tiene un total de:

$$8 \times 2^{10} \times 2 \times 8 \times 2^{10} = 2^{27} \text{ elementos.}$$

Como cada double ocupa 8 bytes, la matriz ocupa en total  $2^{27} \times 2^3 = 2^{30}$  bytes = 1 GB

El código accede a la matriz con el siguiente patrón:

```
m[0][0], m[1][0], ..., m[max-1][0],  
m[0][1], m[1][1], ..., m[max-1][1],  
...  
m[0][2*max-1], m[1][2*max-1], ..., m[max-1][2*max-1]
```

En la primera iteración se realizan  $2^{13}$  accesos. Cada acceso está separado del siguiente  $2^{14}$  elementos, que se corresponden con  $2^{17}$  bytes. Como  $2^{17}$  es múltiplo de  $2^{14}$  (número de bytes por vía), todos los accesos se corresponden con el conjunto 0 de. Por tanto todos los accesos de la primera iteración producen fallo.

Cuando se llega a los primeros elementos de la segunda iteración, los elementos de interés de la iteración anterior han sido todos expulsados de la caché.

Por tanto, todos los accesos producen fallo y la tasa de aciertos es del 0%.

### Apartado C

```
constexpr long max = 8 * 1024; // Constante  
double m[max][2*max];  
// Lectura de valores.  
// ...  
for (int i=0; i<max; ++i) {  
    for (int j=0; j<2*max; ++j) {  
        m[i][j] = 0;  
    }  
}
```

### Apartado D

El código accede a la matriz con el siguiente patrón:



Universidad  
Carlos III de Madrid

Departamento de Informática  
Grado en Ingeniería Informática  
Arquitectura de Computadores

Prueba de evaluación continua  
19 de octubre de 2015



$m[0][0], m[0][1], \dots, m[0][2 \cdot \text{max}-1],$

$m[1][0], m[1][1], \dots, m[1][2 \cdot \text{max}-1],$

...

$m[\text{max}-1][0], m[\text{max}-1][1], \dots, m[\text{max}-1][2 \cdot \text{max}-1],$

En la primera iteración, los 8 primeros accesos se corresponden con el conjunto 0 de la primera vía suponiendo 1 fallo y 7 aciertos. Este patrón se repite a lo largo de todo el acceso a la matriz. Como no se reusan elementos la política de remplazo es irrelevante para el cálculo de la tasa de aciertos que pasa a ser  $7/8$ .



**Ejercicio 3 [2.5 puntos]:** Un determinado procesador tiene un único nivel de memoria caché. La caché de 16 KB es asociativa por conjuntos de 4 vías. El tamaño de línea en todos los niveles es de 64 bytes. Todos los niveles tienen una política de escritura de post-escritura y una política de remplazo LRU.

En esta máquina se está valorando la ejecución del siguiente código:

```
constexpr long max = 8 * 1024; // Constante
double u[max], v[max], w[max], t[max];
// Lectura de valores.
// ...
for (int i=0; j<max; ++i) {
    w[i] = u[i] + v[i]
}
for (int i=0; j<max; ++i) {
    t[i] = u[i] - v[i];
}
```

El vector x se almacena a partir de la dirección de memoria 0x20000000. El resto de vectores se almacenan de forma consecutiva a este.

Se pide:

- Determine cuantos conjuntos tiene la caché en cada vía. [0.5 puntos]
- Determine la tasa de aciertos de la memoria caché para accesos a datos. Considere que los accesos a la constante max y la variable i no generan accesos a memoria. Considere la caché inicialmente vacía. [0.75 puntos]
- Proponga una modificación de código que aplique la técnica de fusión de bucles. [1 punto]
- Determine la tasa de aciertos de la memoria caché para acceso a datos del código modificado. [0.25 puntos]

## SOLUCIÓN:

### Apartado A

Caché L1 (16KB =  $2^{14}$  bytes)

Cada vía tiene  $2^{14}/2^2$  bytes =  $2^{12}$  bytes. Como cada línea tiene 64 bytes ( $2^6$  bytes), el número de conjuntos por línea será  $2^{12}/2^6=2^6$  conjuntos



## Apartado B

El vector  $u$  comienza en la dirección  $2^{29}$  que está alineado al principio de la caché. Cada vector tiene un total de  $2^{13}$  elementos que equivale a  $2^{16}$  bytes. Como cada vía tiene  $2^{12}$  bytes, todos los vectores estarán alineados a principio de caché.

El patrón de accesos es:

$u[0], v[0], w[0], u[1], v[1], w[1], \dots, u[\max], v[\max], w[\max]$

$u[0], v[0], t[0], u[1], v[1], t[1], \dots, u[\max], v[\max], t[\max]$

En el primer bucle se tiene que:

$u[0], v[0]$  y  $w[0]$  generan fallos rellenan la línea del conjunto 0 de las vías 0,1,2.

Posteriormente los accesos para  $i=1..7$  generan aciertos en todos sus accesos. Este patrón se vuelve a repetir hasta que finaliza el bucle. De este modo por cada 3 fallos se producen 21 aciertos.

Para cuando comienza el segundo bucle los primeros elementos del array han sido barridos de la caché, por lo que vuelve a repetirse el mismo patrón ahora con  $u, v$  y  $t$ .

En consecuencia, la tasa de aciertos es  $21/24 = 7/8$

## Apartado C

```
constexpr long max = 8 * 1024; // Constante
double u[max], v[max], w[max], t[max];
// Lectura de valores.
// ...
for (int i=0; i<max; ++i) {
    w[i] = u[i] + v[i]
    t[i] = u[i] - v[i];
}
```

## Apartado D

Ahora el patrón de accesos pasa a ser:



Universidad  
Carlos III de Madrid

Departamento de Informática  
Grado en Ingeniería Informática  
Arquitectura de Computadores

Prueba de evaluación continua  
19 de octubre de 2015



$u[0], v[0], w[0], u[0], v[0], t[0]$

$u[1], v[1], w[1], u[1], v[1], t[1]$

...

En la primera iteración  $u, v, w, t$  pasan a ocupar las cuatro vías del conjunto 0. Para  $i=0$ , se producen 4 fallos y 2 aciertos. Para  $i=1, \dots, 7$  se produce  $6 \times 7 = 42$  aciertos. A partir de aquí el patrón se repite. Por tanto, se producen 4 fallos por cada 44 aciertos.

En consecuencia, la tasa de aciertos es  $44/48 = 11/12$ .



**Ejercicio 3 [2.5 puntos]:** Un determinado procesador tiene un único nivel de memoria caché. La caché de 64 KB es asociativa por conjuntos de 4 vías. El tamaño de línea en todos los niveles es de 64 bytes. Todos los niveles tienen una política de escritura de post-escritura y una política de remplazo LRU.

En esta máquina se está valorando la ejecución del siguiente código:

```
constexpr long max = 8 * 1024; // Constante
struct block {
    int a;
    double b;
    float c;
    long d;
    char e[8];
};
block b[max];
// Lectura de valores.
// ...
for (int i=0; i<max; ++i) {
    b[i].b = 1.0/(double(b[i].a));
}
```

El vector b se almacena a partir de la dirección de memoria 0x20000000. Los tamaños de los tipos de coma flotante son los especificados por IEEE 754. Un int ocupa 4 bytes y un long ocupa 8 bytes.

Asuma que se han seleccionado las opciones del compilador que impiden que se introduzca bytes de relleno entre los campos de la estructura y que por tanto todos los campos se disponen de forma consecutiva en memoria.

Se pide:

- Determine cuantos conjuntos tiene la caché en cada vía. [0.5 puntos]
- Determine la tasa de aciertos a cada nivel de la memoria caché para accesos a datos. Considere que los accesos a la constante max y la variable i no generan accesos a memoria. Considere la caché inicialmente vacía. [0.75 puntos]
- Proponga una modificación de código que elimine el array fusionado y lo sustituya por múltiples arrays paralelos. [1 punto]



- d) Determine la tasa de aciertos de la memoria caché para acceso a datos del código modificado. [0.25 puntos]

## SOLUCIÓN

### Apartado A

Caché L1 ( $64\text{KB} = 2^{16}$  bytes)

Cada vía tiene  $2^{16}/2^2$  bytes =  $2^{14}$  bytes. Como cada línea tiene 64 bytes ( $2^6$  bytes), el número de conjuntos por línea será  $2^{14}/2^6 = 2^8$  conjuntos

### Apartado B

El tamaño de la estructura viene dado por:

- a: int  $\rightarrow$  4 bytes
- b: double  $\rightarrow$  8 bytes
- c: float  $\rightarrow$  4 bytes
- d: long  $\rightarrow$  8 bytes
- e: char[8]  $\rightarrow$  8 bytes

En total un elemento del array ocupará 32 bytes. Por tanto en cada línea de cachés se pueden almacenar dos elementos del array.

El array b comienza en la dirección  $2^{29}$  que está alineada al principio de caché. Cada vector tiene  $2^{13}$  elementos que equivale a  $2^{13} \times 2^5 = 2^{18}$  elementos. Como cada vía tiene  $2^{14}$  bytes, todos los vectores estarán alineados a principio de caché.

El patrón de accesos es:

b[0].a, b[0].b, b[1].a, b[1].b, ...

En la primera iteración se produce un fallo y un acceso en el conjunto 0 de la vía 0. Este patrón se repite durante todo el recorrido del array, por lo que se concluye que la tasa de aciertos es del 50%.

### Apartado C

```
constexpr long max = 8 * 1024; // Constante
```



```
int a[max];  
double b[max];  
float c[max];  
long d[max];  
char e[max][8];  
  
// Lectura de valores.  
// ...  
for (int i=0; j<max; ++i) {  
    b[i] = 1.0/(double(a[i]));  
}
```

#### Apartado D

Ahora el patrón de accesos pasa a ser:

a[0], b[0], a[1], b[1], ...

El array a está alineado a principio de caché (dirección de comienzo  $2^{29}$ ) y tiene un tamaño de  $4 \times 8 \times 1024 = 2^{15}$  que es múltiplo del tamaño de una vía ( $2^{14}$ ). Puesto que todos los arrays tienen un tamaño de elemento de 4 u 8 bytes, todos los arrays tienen su comienzo alineado al comienzo de la caché y tiene un tamaño múltiplo del tamaño de una vía.

Cuando se accede a a[0] se produce un fallo y se rellena una línea con los valores a[0]..a[15], al ser el tamaño de elemento de a de 4 bytes. Cuando se accede a b[0], se rellena la línea de la segunda vía con los valores b[0]...b[7].

Después de estos 2 fallos, para i=1..7, se producen  $2 \times 7 = 14$  aciertos.

Para i=8 el acceso a a[8] sigue siendo un acierto, pero el de b[8] será un fallo. A continuación, los accesos para i=9..15 producen  $2 \times 7 = 14$  aciertos.

En total para los 32 accesos que se producen desde i=0 hasta i=15, se producen 3 fallos y 29 aciertos.

En consecuencia la tasa de fallos será 29/32