Academia CSV
c/Laureano Mediante nº1 Bajo
Tlf 981.28.89.06 movil 685.50.26.05

ESTRUCTURA DE COMPUTADORES
Segunda evaluación (ejercicios)

APELLIDO: _____ **NOMBRE:**_____

[**2.0p**] Un computador tiene una memoria principal de **1 MiB**. Tiene una caché de **2 KiB**, asociativa por conjuntos de **2 vías**, con un tamaño de línea de **16 Bytes** y política de ubicar en escritura (allocate-on-write) y **post-escritura** (write-back). Estando la caché inicialmente vacía, se ejecuta el siguiente código. La matriz m se encuentra almacenada por filas a partir de la dirección de memoria 0x0A000, el resto de variables no provocan accesos a memoria caché, y cada elemento de tipo "int" ocupa exactamente 4 Bytes.

```
int i, j, m[32][256];
for (i = 0; i < 2; i++)
  for (j = 0; j < 2; j++)
    m[j][i*2] = m[j+1][0]
```

Se escriben las modificaciones en cache
y sólo se actualiza MP en caso de reemplazo

1. [**0.3p**] Calcula las direcciones de memoria a las que accede este código.

i	j	lee m[j+1][0]	escribe m[j][i*2]
0	0	m[1][0]	m[0][0]
0	1	m[2][0]	m[1][0]
1	0	m[1][0]	m[0][2]
1	1	m[2][0]	m[1][2]

m[0][0] -> 0x0A000
m[0][2] -> 0x0A000+4*2=0x0A008
m[1][0] -> 0x0A000+256*4=0x0A000+0x400=0x0A400
m[1][2] -> 0x0A400+4*2=0x0A408
m[2][0] -> 0x0A400+0x400=0x0A800

2. [**0.3p**] Determina cómo se divide una dirección física desde el punto de vista de la caché.

1MB=$1*2^{20}$
2KB=$2*2^{10}=2^{11}$
Nº marcos=$2^{11}/2^4 = 2^7$ marcos
$2^7/2$ vias= 2^6 conjuntos

etiqueta	indice/conjunto	desplazamiento
20-6-4=10 bits	6 bits	4 bits

<----------20 bits---------->

3. [**0.4p**] Rellena los campos de la siguiente tabla para los accesos a memoria caché: dirección, etiqueta e índice/conjunto (en **hexadecimal**); si el acceso es un acierto o un fallo; en este último caso, qué tipo de fallo; y si se produce una escritura en memoria principal (MP WR).

Dirección	Etiqueta	Índice	Acierto/Fallo	Tipo de fallo	MP WR
m[1][0] 0x0A400 0000 1010 0100 0000 0000	0x29	0x00	Fallo	Forzoso	No
m[0][0] 0x0A000 0000 1010 0000 0000 0000	0x28	0x00	Fallo	Forzoso	No
m[2][0] 0x0A800 0000 1010 1000 0000 0000	0x2A	0x00	Fallo	Forzoso Reemplaza 0x29	No
m[1][0] 0x0A400 0000 1010 0100 0000 0000	0x29	0x00	Fallo	Capacidad Reemplaza 0x28	sí cuando reemplaza un bloque modificado lo actualiza en MP
m[1][0] 0x0A400 0000 1010 0100 0000 0000	0x29	0x00	Acierto		No
m[0][2] 0x0A008 0000 1010 0000 0000 0100	0x28	0x00	Fallo	Capacidad Reemplaza 0x2A	No
m[2][0] 0x0A800 0000 1010 1000 0000 0000	0x2A	0x00	Fallo	Capacidad Reemplaza 0x29	No
m[1][2] 0x0A408 0000 1010 0100 0000 1000	0x29	0x00	Fallo	Capacidad Reemplaza 0x28	sí

A partir de aquí tengo el problema de que no me dá algoritmo de reemplazamiento por lo tanto voy a suponer LRU.
Se debieron olvidar.

escribe en cache

4. **[0.3p]** Indica cómo quedaría el directorio caché en las líneas ocupadas tras la ejecución de este código

Conjunto	Vía	Bit de validez	Etiqueta
0	0	1	0x29
0	1	1	0x2A

5. **[0.3p]** Sabiendo que el tiempo medio de acceso a memoria durante la ejecución del apartado anterior fue de 50 ciclos, y que el tiempo de acierto caché es de 1 ciclo, calcula el tiempo de acceso a memoria principal.

Tiempo medio de acceso a memoria = Tiempo de acierto + Tasa de fallos × Penalización de fallo

el tiempo de acceso a MP es el tiempo del fallo
tasa de fallos= 7 fallos/8 accesos
50=1+(7/8)*penalización
penalización=tiempo acceso MP=56 ciclos

6. **[0.4p]** Discute la **veracidad** de las siguientes afirmaciones

a) La política de **ubicar en escritura** es mejor tanto para cachés de post-escritura como de escritura directa.

Notas : ubicar en escritura, se carga el bloque en la cache para escribir el dato(lo que siempre hacemos)
post-escritura: solo se escribe en la cache, se actualiza en MP cuando el bloque es reemplazado(lo habitual)
escritura directa: la escritura se hace en la cache y en MP (no es habitual, perdemos la ventaja de la cache)

La politica de ubicar en escritura nos ahorra tiempo porque luego no actualizamos continuamente en MP
que es lo lento por lo cual solo es util con post escritura

b) En este sistema, incorporando una caché de segundo nivel de mayor tamaño se reduciría el tiempo medio de acceso

Sí pues reducimos los accesos a MP que son los lentos