Laboratorio Sesión 05: Repaso de memoria cache

Objetivo

El objetivo de esta sesión es recordar conceptos de memoria cache vistos en EC. Para hacerlo programaréis un simulador de cache básico que simule dos caches de lectura.

Características de la memoria cache

En esta sesión programaremos una memoria cache con las siguientes caractarísticas:

- Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)
- La cache es de sólo lectura (asumiremos que todos los accesos son lecturas)
- La cache será de mapeo directo o 2 asociativa con reemplazo LRU
- Tamaño de la cache: 4 Kbytes
- Tamaño de la línea de cache: 32 bytes

Toma de contacto con el entorno simulador

El simulador se compone de 3 ficheros: `CacheSim.o`, `CacheSim.h` y `MiSimulador.c` El programa principal y algunos componentes del simulador ya están programados y se encuentran en el fichero `CacheSim.o`. Este fichero se encarga de generar las secuencias de test, de imprimir los resultados de la simulación por pantalla con un formato agradable y de comprobar el correcto funcionamiento de vuestro simulador. Antes de que empecéis a programar el simulador, es interesante hacer algunas pruebas con este entorno. Para comenzar, compilad el simulador (`MiSimulador.c` no funciona correctamente, pero compila).

```
$> gcc -m32 CacheSim.o MiSimulador.c tiempo.c -o sim
```

El programa tiene 3 tests:

- Test 0: Genera la secuencia de 20 referencias de la tabla del trabajo previo
- Test 1: Genera accesos secuenciales a un vector de enteros (1000 referencias)
- Test 2: Genera los accesos de un producto de matrices de 25x25 (62500 referencias)

Para pasar cualquiera de los tests, sólo es necesario poner el nº de test como parámetro del simulador. Por ejemplo, para pasar el test 0 escribiríamos:

```
$> sim 0
Test 0 FAIL :-(
$>
```

Evidentemente el test ha fallado, ya que aun no hemos programado el simulador. En caso de que el simulador falle, nos interesará ver qué está pasando. Para ello podemos utilizar la opción `v` (de verbose) en el simulador (la `v` debe aparecer como primer parámetro):

```
$> sim v
eca130 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
eca131 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
ec2172 -> l MP:400136d0 l MC:bffff3e0 TAG:78e530f byte:804822c MISS -> 804816c
...
Test 0 FAIL :-(
```

Esta opción nos dará una salida parecida a la anterior. Como podéis ver las columnas corresponden básicamente a la tabla del ejercicio previo. De esta forma, comparando la salida y la tabla podemos ver dónde está el problema. Dado que en los tests 1 y 2 el número de referencias es muy alto, os recomendamos que no los probéis hasta que os funcione perfectamente el test 0. Con la opción `v`, los tests 1 y 2 se paran tan pronto aparece el primer error para ayudar a su identificación.

Programación del módulo MiSimulador.c

Para programar vuestro simulador de cache tenéis que programar 3 secciones del fichero `MiSimulador.c`:

1. **Estructuras globales** En esta sección tenéis que declarar las estructuras de datos globales necesarias para mantener el estado de la cache. Es necesario que sean globales, ya que la parte principal del simulador es la rutina `reference` que se ejecuta una vez por referencia y, como ya sabéis, su estado desaparece una vez se ejecuta.

2. **Inicialización de la cache** La rutina `init_cache` se llama antes de pasar cada test para inicializar las estructuras de datos globales necesarias. El objetivo es dejar la cache en un estado inicial correcto (cache vacía).

3. **Simulación de referencias** La simulación de las referencias tenéis que hacerla en la rutina `reference`. Esta rutina se llama una vez por cada referencia a simular. Sólo es necesario que generéis el valor correcto de las 7 variables locales que ya tenéis declaradas al inicio de la subrutina y que se corresponden básicamente a las columnas de la tabla del trabajo previo (excepto el booleano `replacement`, que no era necesario en el trabajo previo).

   ```
   void reference (unsigned int address)
   {
     unsigned int byte;
     unsigned int bloque_m;
     unsigned int linea_mc;
     unsigned int via_mc; // esta parámetro sólo se usa en el fichero
                          // MiSimulador2.c para la cache 2 asociativa
     unsigned int tag;
     unsigned int miss; // booleano que indica si es miss
     unsigned int replacement; // booleano que indica si
                               // se reemplaza una línea válida
     unsigned int tag_out; // TAG de la línea reemplazada
   ```

   En otras palabras, lo que tenéis que hacer es implementar el algoritmo que, de forma intuitiva, habéis hecho servir manualmente para rellenar la tabla del estudio previo.

Después de vuestro código, la rutina acaba con una llamada a la rutina `test_and_print` (o `test_and_print2` si es la cache 2 asociativa) para comprobar si los valores de las variables son correctos e imprimirlos por pantalla en caso de tener la opción `v` activada.

Estudio Previo

1. Rellenad la tabla de la hoja de respuestas indicando para cada referencia de la secuencia de referencias la información siguiente (en hexadecimal) para el caso de la cache directa:

   - el byte de la línea a que se accede (byte)
   - el bloque de memoria (bloque M)
   - la línea de memoria cache donde se mapeará la referencia (línea MC)
   - la etiqueta (TAG) que se guardará de esta referencia
   - si el acceso es HIT o MISS,

- y en caso de que se reemplace una línea válida, el TAG de la línea reemplazada (TAG out)

2. Rellenad la tabla de la hoja de respuestas indicando para cada referencia de la secuencia de referencias la información siguiente (en hexadecimal) para el caso de la cache 2 asociativa con reemplazo LRU:
   - el byte de la línea a que se accede (byte)
   - el bloque de memoria (bloque M)
   - el conjunto de memoria cache donde se mapeará la referencia (conj MC)
   - la vía de memoria cache donde se mapeará la referencia (VIA)
   - la etiqueta (TAG) que se guardará de esta referencia
   - si el acceso es HIT o MISS,
   - y en caso de que se reemplace una línea válida, el TAG de la línea reemplazada (TAG out)

   Suponed que el primer acceso en fallo a un determinado conjunto de cache siempre se guarda en la via 0.

3. Dado el siguiente código en C:

```
int vector[1024*10];

for (i=0;i<10240;i++)
   total=vector[i]+i;
```

32 bytes línea cache / 4 bytes (int) = 8 variables int por línea

Cache Directa 1F 31A → 1280F 39680A

Cache Asociativa 1F 31A → 1280F 39680A

10240 / 8 = 1280

Da fallo el primer byte de la primera posición del vector que se coloca en la línea/conjunto del MC

40960 accesos totales

   Calculad cuantos aciertos y fallos en la cache directa obtendremos al ejecutarlo suponiendo que las variables `i` y `total` se almacenan en registros y que la dirección de inicio de la variable `vector` es 0x10f92160.

4. Repetid el cálculo suponiendo que la cache es 2 asociativa con reemplazo LRU.

5. Dado el siguiente código en C:

```
int vector[1024*10];
int vector2[1024*10];

for (i=0;i<10000;i++)
   total=vector[i]+vector2[i]+i;
```

las 2 variables

número de veces que accedemos a cada 1 de las variables

Cache Directa 1F 3A + 1F 3A → 2F * 10000 = 20000F 6A * 10000 = 60000A

Cache Asociativa 2F 62A → 2F * 1250 = 25000F 62A * 1250 = 77500A

Los 2 bloques → 10000/8

num bloques

80000 accesos totales

   Calculad cuantos aciertos y fallos en la cache directa obtendremos al ejecutarlo suponiendo que las variables `i` y `total` se almacenan en registros y que la dirección de inicio de la variable `vector` es 0x10f92160.

6. Repetid el cálculo suponiendo que la cache es 2 asociativa con reemplazo LRU.

Trabajo a realizar durante la Práctica

1. Programad una primera versión del simulador de cache directa en el fichero `MiSimulador.c`. Cuando funcione entregad en el Racó de la asignatura el fichero `MiSimulador.c`.

2. Programad un simulador de la cache 2 asociativa con reemplazo LRU en el fichero `MiSimulador2.c` y probadlo con el programa almacenado en el fichero `CacheSim2.o`. Suponed que el primer acceso en fallo a un determinado bloque de cache siempre se guarda en la via 0. Cuando funcione entregad en el Racó de la asignatura el fichero `MiSimulador2.c`.

3. Medid (usando la opción `test 2`) cuántas instrucciones ejecuta todo el programa en la primera versión que habéis programado.

4. Medid (usando la opción `test 2`) cuántas instrucciones ejecuta todo el programa en la segunda versión que habéis programado.

5. Calculad el número de aciertos y fallos de cache que se obtienen en el `test 2` con la cache directa.

6. Calculad el número de aciertos y fallos de cache que se obtienen en el `test 2` con la cache 2 asociativa.

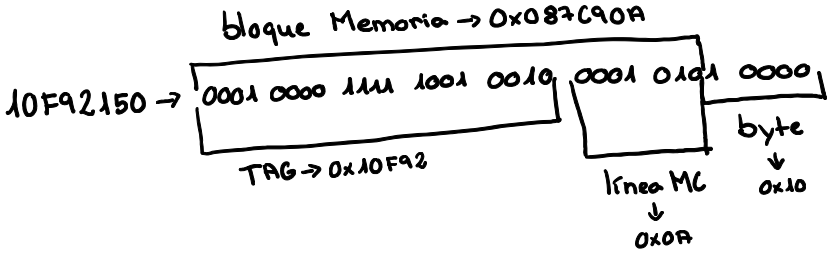
Nombre: _____ Grupo: _____

Nombre: _____

Hoja de respuesta al Estudio Previo

1. Rellenad la siguiente tabla (en hexadecimal):

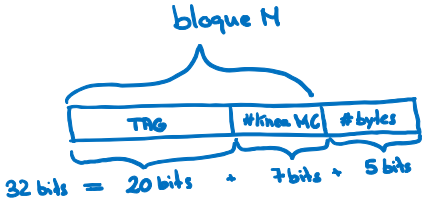
@	byte	bloque M	línea MC	TAG	HIT/MISS	TAG out
10f92150	10	087C90A	0A	10F92	MISS	
10f92151	11	087C90A	0A	10F92	HIT	
10f8a192	12	087C50C	0C	10F8A	MISS	
10f92153	13	087C90A	0A	10F92	HIT	
10f8b195	15	087C58C	0C	10F8B	MISS	10F8A
10f8b195	15	087C58C	0C	10F8B	HIT	
10f93156	16	087C98A	0A	10F93	MISS	10F92
10f92157	17	087C90A	0A	10F92	MISS	10F93
10f8a198	18	087C50C	0C	10F8A	MISS	10F8B
10f93159	19	087C98A	0A	10F93	MISS	10F92
12f92250	10	097C912	12	12F92	MISS	
10f92151	11	087C90A	0A	10F92	MISS	10F93
10f8a192	12	087C50C	0C	10F8A	HIT	
12f92253	13	097C912	12	12F92	HIT	
10f8b195	15	087C58C	0C	10F8B	MISS	10F8A
10f8b195	15	087C58C	0C	10F8B	HIT	
10f93156	16	087C98A	0A	10F93	MISS	10F92
12f92257	17	097C912	12	12F92	HIT	
10f8a298	18	087C514	14	10F8A	MISS	
10f93159	19	087C98A	0A	10F93	HIT	



Tamaño cache = 4Kbytes = 4096 bytes

4096 / 32 = 128 líneas MC → $\log_2(128) = 7$ bits para identificar la línea de la MC que queremos consultar

32 bytes por línea ⇒ #bytes → $\log_2(32) = 5$ bits para identificar el byte deseado



2. Rellenad la siguiente tabla (en hexadecimal):

@	byte	bloque M	conj MC	VIA	TAG	HIT/MISS	TAG out
10f92150	10	087C90A	0A	0	021F24	MISS	
10f92151	11	087C90A	0A	0	021F24	HIT	
10f8a192	12	087C50C	0C	0	021F14	MISS	
10f92153	13	087C90A	0A	0	021F24	HIT	
10f8b195	15	087C58C	0C	1	021F16	MISS	
10f8b195	15	087C58C	0C	1	021F16	HIT	
10f93156	16	087C98A	0A	1	021F26	MISS	
10f92157	17	087C90A	0A	0	021F24	HIT	
10f8a198	18	087C50C	0C	0	021F14	HIT	
10f93159	19	087C98A	0A	1	021F26	HIT	
12f92250	10	097C912	12	0	025F24	MISS	
10f92151	11	087C90A	0A	0	021F24	HIT	
10f8a192	12	087C50C	0C	0	021F14	HIT	
12f92253	13	097C912	12	0	025F24	HIT	
10f8b195	15	087C58C	0C	0	021F16	HIT	
10f8b195	15	087C58C	0C	0	021F16	HIT	
10f93156	16	087C98A	0A	0	021F26	HIT	
12f92257	17	097C912	12	1	025F24	HIT	
10f8a298	18	087C514	14	0	021F14	MISS	
10f93159	19	087C98A	0A	0	021F26	HIT	

Patrón 1F 31A

3. Para el primer código C, la cache directa obtiene:
   Aciertos: 39 680 Fallos: 1280

4. Para el primer código C, la cache 2 asociativa con reemplazo LRU obtiene:
   Aciertos: 39 680 Fallos: 1280

1F 31A

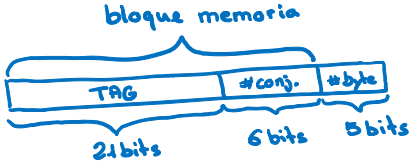
5. Para el segundo código C, la cache directa obtiene:
   Aciertos: 60 000 Fallos: 20000

16F 48A

6. Para el segundo código C, la cache 2 asociativa con reemplazo LRU obtiene:
   Aciertos: 77 500 Fallos: 2500

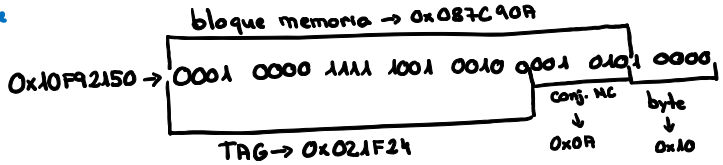
2F 62A

4096/32/2 = 64 conjuntos MC
(2 ↑ 2-asociativa)



$\log_2(64) = 6$ bits para identificar el conjunto

$\log_2(32) = 5$ bits para identificar el byte dentro de la línea



Nombre:_____ Grupo:_____

Nombre:_____

Hoja de respuestas de la práctica

1. La primera versión (cache directa) funciona correctamente (S/N): S
2. La segunda versión (cache 2 asociativa) funciona correctamente (S/N): S
3. Con la primera versión de la rutina (cache directa), el programa completo ejecuta: 8.681.595 instrucciones con la opción `test 2`. valgrind --tool=lackey ./sim 2
4. Con la segunda versión de la rutina (cache 2 asociativa), el programa completo ejecuta: 10.307.354 instrucciones con la opción `test 2`. valgrind --tool=lackey ./sim 2
5. Calculad el número de aciertos y fallos de cache que se obtienen en el `test 2` con la cache directa:
   Aciertos: 58272 Fallos: 4228
6. Calculad el número de aciertos y fallos de cache que se obtienen en el `test 2` con la cache 2 asociativa:
   Aciertos: 59432 Fallos: 3068
7. Recordad entregar los ficheros `MiSimulador.c` y `MiSimulador2.c` en el Racó de la asignatura. Debéis entregar sólo los dos ficheros fuentes, sin comprimir ni cambiarles el nombre, y sólo una versión por pareja de laboratorio (es indistinto que miembro de la pareja entregue).