

Laboratorio Sesión 06: Políticas de escritura

Objetivo

El objetivo de estas sesiones es asentar los conocimientos sobre las diferentes políticas de escritura de las memorias cache. Para hacerlo programaréis un simulador de cache básico que simule lecturas y escrituras.

Características de la memoria cache

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

- Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)
- La cache será de mapeo directo
- Tamaño de la cache: 4 Kbytes
- Tamaño de la línea de cache: 32 bytes
- Política de escritura: Write Through
- Política de emplazamiento: Write No Allocate

Toma de contacto con el entorno simulador

El simulador se compone de 3 ficheros: `CacheSimWT.o`, `CacheSim.hy` y `MiSimulador.c`. El programa principal y algunos componentes del simulador ya están programados y se encuentran en el fichero `CacheSimWT.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-m32CacheSimWT.oMiSimulador.ctiempo.c-osim
```

El programa tiene 3 tests:

- Test0: Generala secuencia de 20 referencias de la tabla del trabajo previo
- Test1: Genera accesos secuenciales a un vector de enteros (1000 referencias)
- Test2: Generala accesos de un producto de matrices de 25x25 (62500 referencias)

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

```
$>sim0
Test0 FAIL:- (
$>
```

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

```
$>simv0
eca130L->1MP:4212dfa01MC:bfffe7a0TAG:804a000byte:40007dcfMISSL:134513233E:1073774031->fd0    eca131E-
>1MP:4212dfa01MC:bfffe7a0TAG:804a000byte:40007dcfMISSL:134513233E:1073774031->fd0    ec2172E-
>1MP:4212dfa01MC:bfffe7a0TAG:804a000byte:40007dcfMISSL:134513233E:1073774031->fd0
...
Test0 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 cachete 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 caché. Es necesario que se declaren 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 caché La rutina `init_cache` se llama antes de pasar cada test para inicializar las estructuras de datos globales necesarias. El objetivo es dejar la caché en un estado inicial correcto (caché 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. Solo es necesario que generéis el valor correcto de las 11 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 LE)
{
    unsigned int byte;
    unsigned int bloque_m;
    unsigned int linea_mc;
    unsigned int tag;
    unsigned int miss; // booleano que indica si es miss
    unsigned int lec_mp; // booleano que indica si se lee de MP
    unsigned int mida_lec_mp;
    unsigned int esc_mp; // booleano que indica si se escribe en MP
    unsigned int mida_esc_mp;
    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` para comprobar si los valores de las variables son correctos e imprimirlos por pantalla en caso de tener la opción `v` activada.

4. Impresión de resultados Si tenéis que imprimir algún resultado final, la rutina `final` se ejecuta una vez al finalizar el programa.

Estudio Previo

1. Rellenar la tabla de la hoja de respuestas indicando, para cada referencia de la secuencia de referencias, la información siguiente (en hexadecimal):
 - el byte de la línea a que se accede (byte) ■
 - el bloque de memoria (bloque M)
 - la línea de memoria caché donde se mapará la referencia (línea MC)

- la etiqueta (TAG) que se guardará de esta referencia
- si el acceso es HIT o MISS,
- si hay una lectura de memoria principal, indicad la cantidad de bytes leídos (lec MP),
- si hay una escritura en memoria principal, indicad la cantidad de bytes escritos (esc MP),
- y en caso de que se reemplaze una línea inválida, el TAG de la línea reemplazada (TAG out).

La memoria cachetiene las siguientes características:

- Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)
- La cache será de mapeo directo
- Tamaño de la cache: 4 Kbytes
- Tamaño de la línea de cache: 32 bytes
- Política de escritura: Write Through
- Política de emplazamiento: Write NO Allocate

2. Rellenad la tabla de la hoja de respuestas indicando, para cada referencia de la secuencia de referencias, la información siguiente (en hexadecimal):

- el byte de la línea a que se accede (byte)
- el bloque de memoria (bloque M)
- la línea de memoria caché donde se mapeará la referencia (línea MC)
- la etiqueta (TAG) que se guardará de esta referencia
- si el acceso es HIT o MISS,
- si hay una lectura de memoria principal, indicad la cantidad de bytes leídos (lec MP),
- si hay una escritura en memoria principal, indicad la cantidad de bytes escritos (esc MP),
- y en caso de que se reemplaze una línea inválida, el TAG de la línea reemplazada (TAG out).

La memoria cachetiene las siguientes características:

- Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)
- La cache será de mapeo directo
- Tamaño de la cache: 4 Kbytes
- Tamaño de la línea de cache: 32 bytes
- Política de escritura: Copy Back
- Política de emplazamiento: Write Allocate

Trabajo a realizar durante la Práctica

1. Programad una versión del simulador de cache de lectura/escritura con política Write Through + Write NO Allocate y comprobad su correcto funcionamiento. Cuando funcione entregad el fichero `MiSimulador.c` en el Racó de la asignatura.
2. Implementad también un simulador con el mismo funcionamiento pero con política de escritura Copy Back + Write Allocate en el fichero `MiSimulador2.c`. Para comprobar su funcionamiento utilizad el fichero `CacheSimCB.o`. Cuando funcione entregad el fichero `MiSimulador2.c` en el Racó de la asignatura.

3. Modificad vuestros simuladores de cache para contar cuántos accesos aciertan y fallan en caché en una ejecución del programa. Podéis imprimir el resultado poniendo el código en la rutina `final`. Averiguad qué política de escritura es mejor para los accesos que ejecuta el `test 2`.
4. Recordad entregar los ficheros `MiSimulador.cy` y `MiSimulador2.c` en el `Racó de la signatura`. 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).

Nombre: Liang Liang Chen Xu

Grupo: 12(2)

Nombre: _____

Hoja de respuesta al Estudio Previo

1. La memoria cache tiene las siguientes características:

■ Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)

■ La cache será de mapeo directo ■

Tamaño de la cache: 4 Kbytes

■ Tamaño de la línea de cache: 32 bytes ■

Política de escritura: Write Through

■ Política de emplazamiento: Write No Allocate Rellenado

la siguiente tabla (en hexadecimal):

@	L/E	byte	bloque M	línea MC	TAG	H/M	lecMP	escMP	TAGout
00eca130	L	10	76509	09	00ECA	M	32	-	-
00eca131	E	11	76509	09	00ECA	H	-	1	-
00ec2172	E	12	7610B	0B	00EC2	M	-	1	-
00eca133	L	13	76509	09	00ECA	H	-	-	-
00ec3175	L	15	7618B	0B	00EC3	M	32	-	-
00ec3175	L	15	7618B	0B	00EC3	H	-	-	-
00ecb136	L	16	76589	09	00ECB	M	32	-	00ECA
00eca137	E	17	76509	09	00ECA	M	-	1	-
00ec2178	L	18	7610B	0B	00EC2	M	32	-	00EC3
00ecb139	E	19	76589	09	00ECB	H	-	1	-
10eca230	L	10	876511	11	10ECA	M	32	-	-
00eca131	E	11	76509	09	00ECA	M	-	1	-
00ec2172	L	12	7610B	0B	00EC2	H	-	-	-
10eca233	E	13	876511	11	10ECA	H	-	1	-
00ec3175	E	15	7618B	0B	00EC3	M	-	1	-
00ec3175	L	15	7618B	0B	00EC3	M	32	-	00EC2
00ecb136	L	16	76589	09	00ECB	H	-	-	-
10eca237	L	17	876511	11	10ECA	H	-	-	-
00ec2278	L	18	76113	13	00EC2	M	32	-	-
00ecb139	L	19	76589	09	00ECB	H	-	-	-

2. La memoria cachetiene las siguientes características:

- Las direcciones son de 32 bits (para simplificar asumiremos que todos los accesos son a bytes)
- La cache será de mapeo directo ■
- Tamaño de la cache: 4 Kbytes
- Tamaño de la línea de cache: 32 bytes ■
- Política de escritura: Copy Back
- Política de emplazamiento: Write Allocate

Rellenad la siguiente tabla (en hexadecimal):

@	L/E	byte	bloque M	línea MC	TAG	H/M	lecMP	escMP	TAGout
00eca130	L	10	76509	09	00ECA	M	32	-	-
00eca131	E	11	76509	09	00ECA	H	-	-	-
00ec2172	E	12	7610B	0B	00EC2	M	32	-	-
00eca133	L	13	76509	09	00ECA	H	-	-	-
00ec3175	L	15	7618B	0B	00EC3	M	32	32	00EC2
00ec3175	L	15	7618B	0B	00EC3	H	-	-	-
00ecb136	L	16	76589	09	00ECB	M	32	32	00ECA
00eca137	E	17	76509	09	00ECA	M	32	-	00ECB
00ec2178	L	18	7610B	0B	00EC2	M	32	-	00EC3
00ecb139	E	19	76589	09	00ECB	M	32	32	00ECA
10eca230	L	10	876511	11	10ECA	M	32	-	-
00eca131	E	11	76509	09	00ECA	M	32	32	00ECB
00ec2172	L	12	7610B	0B	00EC2	H	-	-	-
10eca233	E	13	876511	11	10ECA	H	-	-	-
00ec3175	E	15	7618B	0B	00EC3	M	32	-	00EC2
00ec3175	L	15	7618B	0B	00EC3	H	-	-	-
00ecb136	L	16	76589	09	00ECB	M	32	32	00ECA
10eca237	L	17	876511	11	10ECA	H	-	-	-
00ec2278	L	18	76113	13	00EC2	M	32	-	-
00ecb139	L	19	76589	09	00ECB	H	-	-	-