Laboratorio Sesi´on 05: Repaso de memoria cache

# Objetivo

El objetivo de esta sesi´on es recordar conceptos de memoria cache vistos en EC. Para hacerlo programar´eis un simulador de cache b´asico que simule dos caches de lectura.

# Caracter´ısticas de la memoria cache

En esta sesi´on programaremos una memoria cache con las siguientes caractar´ısticas:

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

La cache es de so´lo lectura (asumiremos que todos los accesos son lecturas)

La cache ser´a de mapeo directo o 2 asociativa con reemplazo LRU

Taman˜o de la cache: 4 Kbytes

Taman˜o de la l´ınea 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 esta´n programados y se encuen- tran en el fichero CacheSim.o. Este fichero se encarga de generar las secuencias de test, de imprimir los resultados de la simulaci´on por pantalla con un formato agradable y de compro- bar el correcto funcionamiento de vuestro simulador. Antes de que empec´eis 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, so´lo es necesario poner el no de test como para´metro del simulador. Por ejemplo, para pasar el test 0 escribir´ıamos:

$> 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 interesar´a ver qu´e esta´ pasando. Para ello podemos utilizar la opci´on v (de verbose) en el simulador (la v debe aparecer como primer para´metro):

$> 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 opci´on nos dara´ una salida parecida a la anterior. Como pod´eis ver las columnas corres- ponden b´asicamente a la tabla del ejercicio previo. De esta forma, comparando la salida y la tabla podemos ver d´onde esta´ el problema. Dado que en los tests 1 y 2 el nu´mero de referen- cias es muy alto, os recomendamos que no los prob´eis hasta que os funcione perfectamente el test 0. Con la opci´on v, los tests 1 y 2 se paran tan pronto aparece el primer error para ayudar a su identificaci´on.

# Programacio´n del m´odulo MiSimulador.c

Para programar vuestro simulador de cache ten´eis que programar 3 secciones del fichero

MiSimulador.c:

1. Estructuras globales En esta secci´on ten´eis 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 sab´eis, su estado desaparece una vez se ejecuta.
2. Inicializaci´on 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 vac´ıa).
3. Simulaci´on de referencias La simulaci´on de las referencias ten´eis que hacerla en la rutina reference. Esta rutina se llama una vez por cada referencia a simular. S´olo es necesario que gener´eis el valor correcto de las 7 variables locales que ya ten´eis declaradas al inicio de la subrutina y que se corresponden b´asicamente 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 par´ametro s´olo 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 l´ınea v´alida unsigned int tag\_out; // TAG de la l´ınea reemplazada

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

Despu´es de vuestro c´odigo, 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 opci´on v activada.

# Estudio Previo

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

el byte de la l´ınea a que se accede (byte) el bloque de memoria (bloque M)

la l´ınea de memoria cache donde se mapeara´ la referencia (l´ınea MC) la etiqueta (TAG) que se guardara´ de esta referencia

si el acceso es HIT o MISS,

y en caso de que se reemplace una l´ınea va´lida, el TAG de la l´ınea reemplazada (TAG out)

1. Rellenad la tabla de la hoja de respuestas indicando para cada referencia de la secuencia de referencias la informaci´on siguiente (en hexadecimal) para el caso de la cache 2 asociativa con reemplazo LRU:

el byte de la l´ınea a que se accede (byte) el bloque de memoria (bloque M)

el conjunto de memoria cache donde se mapeara´ la referencia (conj MC) la v´ıa de memoria cache donde se mapeara´ la referencia (VIA)

la etiqueta (TAG) que se guardara´ de esta referencia si el acceso es HIT o MISS,

y en caso de que se reemplace una l´ınea va´lida, el TAG de la l´ınea reemplazada (TAG out)

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

1. Dado el siguiente c´odigo en C:

int vector[1024\*10]; for (i=0;i<10240;i++)

total=vector[i]+i;

Calculad cuantos aciertos y fallos en la cache directa obtendremos al ejecutarlo supo- niendo que las variables i y total se almacenan en registros y que la direcci´on de inicio de la variable vector es 0x10f92160.

1. Repetid el c´alculo suponiendo que la cache es 2 asociativa con reemplazo LRU.
2. Dado el siguiente c´odigo en C:

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

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

Calculad cuantos aciertos y fallos en la cache directa obtendremos al ejecutarlo supo- niendo que las variables i y total se almacenan en registros y que la direcci´on de inicio de la variable vector es 0x10f92160.

1. Repetid el c´alculo suponiendo que la cache es 2 asociativa con reemplazo LRU.

# Trabajo a realizar durante la Pr´actica

1. Programad una primera versi´on del simulador de cache directa en el fichero MiSimulador.c. Cuando funcione entregad en el Raco´ 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 Raco´ de la asignatura el fichero MiSimulador2.c.
3. Medid (usando la opci´on test 2) cua´ntas instrucciones ejecuta todo el programa en la primera versi´on que hab´eis programado.
4. Medid (usando la opci´on test 2) cua´ntas instrucciones ejecuta todo el programa en la segunda versi´on que hab´eis programado.
5. Calculad el nu´mero de aciertos y fallos de cache que se obtienen en el test 2 con la cache directa.
6. Calculad el nu´mero 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 | 87c90a | a | 10f92 | miss |  |
| 10f92151 | 11 | 87c90a | a | 10f92 | hit |  |
| 10f8a192 | 12 | 87c50c | c | 10f8a | miss |  |
| 10f92153 | 13 | 87c90a | a | 10f92 | hit |  |
| 10f8b195 | 15 | 87c58c | c | 10f8b | miss | 10f8a |
| 10f8b195 | 15 | 87c58c | c | 10f8b | hit |  |
| 10f93156 | 16 | 87c98a | a | 10f93 | miss | 10f92 |
| 10f92157 | 17 | 87c90a | a | 10f92 | miss | 10f93 |
| 10f8a198 | 18 | 87c50c | c | 10f8a | miss | 10f8b |
| 10f93159 | 19 | 87c98a | a | 10f93 | miss | 10f92 |
| 12f92250 | 10 | 97c912 | 12 | 12f92 | miss |  |
| 10f92151 | 11 | 87c90a | a | 10f92 | miss | 10f93 |
| 10f8a192 | 12 | 87c50c | c | 10f8a | hit |  |
| 12f92253 | 13 | 97c912 | 12 | 12f92 | hit |  |
| 10f8b195 | 15 | 87c58c | c | 10f8b | miss | 10f8a |
| 10f8b195 | 15 | 87c58c | c | 10f8b | hit |  |
| 10f93156 | 16 | 87c98a | a | 10f93 | miss | 10f92 |
| 12f92257 | 17 | 97c912 | 12 | 12f92 | hit |  |
| 10f8a298 | 18 | 87c514 | 14 | 10f8a | miss |  |
| 10f93159 | 19 | 87c98a | a | 10f93 | hit |  |

1. Rellenad la siguiente tabla (en hexadecimal):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| @ | byte | bloque M | conj MC | VIA | TAG | HIT/MISS | TAG out |
| 10f92150 | 10 | 87c90a | a | 0 | 10f92 | miss |  |
| 10f92151 | 11 | 87c90a | a | 0 | 10f92 | hit |  |
| 10f8a192 | 12 | 87c50c | c | 0 | 10f8a | miss |  |
| 10f92153 | 13 | 87c90a | a | 0 | 10f92 | hit |  |
| 10f8b195 | 15 | 87c58c | c | 1 | 10f8b | miss |  |
| 10f8b195 | 15 | 87c58c | c | 1 | 10f8b | hit |  |
| 10f93156 | 16 | 87c98a | a | 1 | 10f93 | miss |  |
| 10f92157 | 17 | 87c90a | a | 0 | 10f92 | hit |  |
| 10f8a198 | 18 | 87c50c | c | 0 | 10f8a | hit |  |
| 10f93159 | 19 | 87c98a | a | 1 | 10f93 | hit |  |
| 12f92250 | 10 | 97c912 | 12 | 0 | 12f92 | miss |  |
| 10f92151 | 11 | 87c90a | a | 0 | 10f92 | hit |  |
| 10f8a192 | 12 | 87c50c | c | 0 | 10f8a | hit |  |
| 12f92253 | 13 | 97c912 | 12 | 0 | 12f92 | hit |  |
| 10f8b195 | 15 | 87c58c | c | 1 | 10f8b | hit |  |
| 10f8b195 | 15 | 87c58c | c | 1 | 10f8b | hit |  |
| 10f93156 | 16 | 87c98a | a | 1 | 10f93 | hit |  |
| 12f92257 | 17 | 97c912 | 12 | 0 | 12f92 | hit |  |
| 10f8a298 | 18 | 87c514 | 14 | 0 | 10f8a | miss |  |
| 10f93159 | 19 | 87c98a | a | 1 | 10f93 | hit |  |

1. Para el primer c´odigo C, la cache directa obtiene:

Aciertos: Fallos:

1. Para el primer c´odigo C, la cache 2 asociativa con reemplazo LRU obtiene:

Aciertos:

Fallos:

1. Para el segundo c´odigo C, la cache directa obtiene:

Aciertos:

Fallos:

1. Para el segundo c´odigo C, la cache 2 asociativa con reemplazo LRU obtiene:

Aciertos: ss

Fallos:

Nombre: Grupo:

Nombre:

# Hoja de respuestas de la pr´actica

1. La primera versi´on (cache directa) funciona correctamente (S/N):
2. La segunda versi´on (cache 2 asociativa) funciona correctamente (S/N):
3. Con la primera versi´on de la rutina (cache directa), el programa completo ejecuta: instrucciones con la opci´on test 2.
4. Con la segunda versi´on de la rutina (cache 2 asociativa), el programa completo ejecuta: instrucciones con la opci´on test 2.
5. Calculad el nu´mero de aciertos y fallos de cache que se obtienen en el test 2 con la cache directa:

Aciertos: Fallos:

1. Calculad el nu´mero de aciertos y fallos de cache que se obtienen en el test 2 con la cache 2 asociativa:

Aciertos: Fallos:

1. Recordad entregar los ficheros MiSimulador.c y MiSimulador2.c en el Raco´ de la asignatura. Deb´eis entregar so´lo los dos ficheros fuentes, sin comprimir ni cambiarles el nombre, y so´lo una versi´on por pareja de laboratorio (es indistinto que miembro de la pareja entregue).