Simulación de sistemas de memoria con Xcache32

1. Introducción

En este tema simularemos el comportamiento de un sistema de memoria que se compone de una jerarquía de dos niveles: una memoria principal y una memoria caché. La caché es una memoria más pequeña y rápida que la principal y que se sitúa entre ésta y el procesador con el fin de acelerar el tiempo de acceso a memoria. El programa a ejecutar se caracterizará por su traza o secuencia de petición de direcciones. La simulación permite seleccionar el tamaño de la caché y de sus líneas, así como algunas de las políticas de gestión de la caché que se implementan en microprocesadores reales.

2. Simuladores dinero y Xcache32

El programa Xcache32 es un interfaz amigable para Windows que nos permite la simulación de este tipo de sistemas, ofreciendo de forma gráfica la evolución de los algoritmos. Xcache32 realiza la simulación mediante la llamada al programa `dinero.exe` que corre bajo MSDOS.

El punto de partida de la simulación lo constituye un fichero que contiene la traza de la ejecución de un programa. Esta traza, nos muestra las referencias a memoria que el programa efectuó durante su ejecución. El fichero de traza para el programa dinero se compone de líneas de texto que constituyen referencias a memoria. Estas líneas constan de dos números que indican, respectivamente, el tipo de referencia y la dirección de memoria a la cual se hace referencia. Los ficheros de traza se guardan con la extensión `.din`.

Existen cinco tipos de referencias a memoria. Cada uno de ellos se identifica por un valor según se indica en la siguiente tabla:

Tipo de referencia	Identificador
Lectura de datos	0
Escritura de datos	1
Búsqueda de instrucción	2
Acceso desconocido	3
Limpieza (flush) de caché	4

Se puede acceder a direcciones de memoria comprendidas en el rango `0 - FFFFFFFF H`. Las palabras de memoria se direccionan por bytes. Un fragmento de archivo de traza sería por ejemplo:

```
0 000000
1 000000
```

```
1 000001
1 000002
2 0000A4
2 0000A5
```

donde el primer número es el tipo de operación y el segundo la dirección a la que se accede.

El procedimiento para realizar la simulación de la traza de un programa sería:

1. En la opción SIMULATION elegir RUN DINERO. Aquí indicaremos el archivo de entrada (.din), el archivo de salida (.out) y los parámetros de la caché: tamaño, algoritmo de reemplazo, ...
2. Una vez ejecutado dinero, tenemos dos opciones: ver el resultado global de la simulación (FILE - OPEN DINERO OUTPUT FILE), o bien ejecutar paso a paso la simulación (FILE - OPEN STREAM INPUT FILE). En el primer caso, hay que abrir el fichero `.out`, y el el segundo, el fichero `.din`.

En la ejecución paso a paso, la aplicación nos presenta cinco ventanas donde se muestran, respectivamente, la memoria principal, la caché, la traza de ejecución, diferentes estadísticos e información de tiempo de ejecución.

En caso de elegir cachés separadas para datos e instrucciones, las ventanas de memoria y de caché se desdoblan en dos para mostrar el acceso a instrucciones y datos por separado.

En la ventana de memoria principal (data/instruction memory status) se representa en cada casilla una palabra de memoria seguida de su ubicación. A medida que realizamos la simulación, veremos cómo son leídas y/o escritas las diferentes posiciones.

En la ventana de caché (data/instruction cache), se representan los diferentes bloques (líneas) de las que consta la caché. Cada bloque o línea almacenará varias palabras de memoria principal. Estas líneas se organizan por conjuntos. Un conjunto se compone de varios bloques a los que se accede de forma asociativa.

3. Especificación de la caché a simular

Para determinar la memoria caché que queremos simular hay que proporcionar dos tipos de información al simulador: Los **parámetros**, que determinan el tamaño de cada uno de los elementos de la caché, y las **políticas**, que indican la estrategia a seguir durante la búsqueda, actualización y reemplazo de sus datos.

3.1. Parámetros

Una caché está organizada en líneas, que son los bloques de información mínima que se transfieren entre caché y memoria principal. El tamaño de la línea indicará el número de palabras que hay en ella, y a su vez, cada palabra se compondrá de un número de bytes.

En general, el simulador tiene una serie de parámetros fijos y otros que el usuario puede seleccionar. Son los siguientes:

- **Parámetros fijos:**

  - El tamaño de la memoria principal, $2^n bytes$. Al ser la memoria direccionable byte a byte en el rango desde 0h a FFFFFFFFh, esto nos dá un valor de $n = 32$ y una memoria de 2^{32} bytes (4 Gigabytes).
  - El tamaño de la palabra de memoria, 2^b bytes. Con palabras de 32 bits como las consideradas aquí, `b=2` y la memoria principal tiene un total de 2^{30} palabras.

- **Parámetros configurables:**

  - El tamaño de la memoria caché, 2^t bytes.
  - El tamaño de la línea de caché, 2^w palabras.
  - El nivel de asociatividad, 2^m, o número de líneas que tiene cada uno de los conjuntos en los que puede organizarse la memoria caché. Este parámetro determina también el número de conjuntos, 2^c, en base a la siguiente fórmula:

$$\boxed{2^c = \frac{2^t}{2^b \cdot 2^w \cdot 2^m}}$$

3.2. Políticas

3.2.1. Organización (*general options & sizes*)

Para agilizar el proceso de búsqueda de una palabra en la caché, ésta suele implementarse mediante una memoria asociativa en la que la dirección a localizar se compara simultáneamente con la dirección base de cada una de las líneas, obteniendo de inmediato la línea buscada.

Sin embargo, el alto coste de una memoria asociativa hace que la organización más utilizada no sea **totalmente asociativa**, sino **asociativa por conjuntos**, donde la caché se divide en 2^c conjuntos de 2^m líneas cada uno. A partir de la dirección de memoria solicitada, se obtiene de forma directa el conjunto de la caché que tiene asignado, y dentro del conjunto se busca ahora asociativamente la línea correspondiente.

Si tenemos conjuntos de una sola línea, entonces se dice que la caché tiene una organización **directa**, ya que la línea se obtiene directamente a partir de la dirección de memoria. Para indicar esta opción al simulador, seleccionaremos un nivel de asociatividad, 2^m, igual a la unidad (m=0). Por el contrario, para especificar una caché totalmente asociativa, seleccionaremos un nivel de asociatividad que vendrá determinado por los valores de `b`, `t` y `w`, según se indica en la siguiente fórmula:

$$\boxed{2^m = \frac{2^t}{2^w \cdot 2^b}}$$

3.2.2. Escritura

Cuando se solicita una operación de escritura a memoria de una celda cuyo valor se encuentra en caché, hay dos formas básicas de proceder (*write policy*):

- **Escritura directa** o *write-through*: El valor se actualiza en caché y en memoria principal de manera simultánea.
- **Post-escritura** o *write-back*: El valor se actualiza únicamente en la caché y la memoria principal se actualiza cuando esa línea sea reemplazada por otra en la caché. Esta estrategia es más rápida que la anterior, pero a costa de no asegurar la consistencia de los datos de caché con sus homólogos de memoria principal.

Si se realiza un acceso de escritura, y el dato buscado no está en cache, hay dos posibilidades (*write allocation policy*):

- *Write allocate*: similar a la lectura, el bloque con el dato se carga en cache. Suele ir asociado a la política de *write-back*.
- *Non write allocate*: el bloque se modifica directamente en memoria principal y no se carga en cache. Suele ir asociado a la política de *write-through*.

3.2.3. Reemplazo (*replacement policy*)

El objetivo de una memoria caché consiste en almacenar las palabras de memoria que más se referencian, con el fin de maximizar el índice de aciertos a ésta.

Cuando se accede a una palabra que no está en caché, puede introducirse en ella reemplazando a otra que ya estaba. La línea a sustituir se selecciona mediante una política de reemplazo. Los criterios que más se utilizan para este propósito son:

- Random: Se reemplaza una línea al azar. Es el criterio menos eficaz, pero el más barato de implementar.
- LRU (Least Recently Used): Se reemplaza la línea menos recientemente utilizada. Es el método más eficiente, pues está basado en los *principio de localidad espacial y temporal* que caracterizan la secuencia de direcciones solicitadas por un programa. No obstante, resulta también muy caro de implementar.
- FIFO (First In First Out): Se reemplaza la línea que más tiempo ha permanecido en la caché, independientemente de cuánto o cuándo se haya utilizado. Es un compromiso de coste y eficiencia intermedios respecto a las dos anteriores.

3.2.4. Precarga (*fetch policy*)

Las técnicas de precarga tratan de maximizar el índice de aciertos a caché por medio de una anticipación, es decir, introduciendo las palabras de memoria en caché antes de que sean solicitadas por el procesador.

Las políticas de precarga están también basadas en los principios de localidad espacial y temporal. Las más utilizadas son:

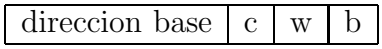
- **Siempre:** Cuando se solicita una dirección de memoria principal, se lleva a caché su línea y la(s) siguiente(s).

- **Bajo fallo:** Se carga en caché la(s) líneas siguiente(s) sólo si la dirección solicitada produjo un fallo en la caché.
- **Por demanda:** No se efectúa ningún tipo de precarga.

4. Visualización de la simulación

Una vez seleccionados los distintos parámetros y alternativas de diseño del sistema caché, procederemos a la ejecución de la simulación en sí.

El simulador muestra cada una de las direcciones de memoria accedidas en base al siguiente formato:



donde con `c`, `w` y `b` calculamos 2^c conjuntos, 2^w palabras por línea de caché y 2^b bytes por palabra, respectivamente, según se ha explicado en el apartado 3.1.

Además, el simulador mantiene información para cada línea de caché, la cual puede visualizarse pulsando dos veces sobre la línea. Esta información se compone de:

- La dirección base de la línea, que se usa para localizarla en la búsqueda asociativa que se realiza dentro del conjunto en el que se encuentra.
- El contenido de las 2^w palabras de que se compone la línea.
- Información de acceso que permite saber si la línea ha sido leída y/o escrita.

A medida que avanza la simulación, las posiciones de la caché se van marcando con diversos colores, en función del tipo de suceso que ocurrió sobre cada una de esas posiciones la última vez que fueron referenciadas. Dichos colores nos indican:

- **Línea no ocupada**: no se ha cargado nada aún en esa línea.
- **Éxito en el acceso** (*cache hit*): La palabra accedida se encontraba con un valor válido en la caché.
- **Línea no valida** (*invalid*): porque en esa línea se ha producido una escritura y aparece un problema de inconsistencia de datos con la memoria principal.
- **Fallo de caché**: los motivos de los fallos de caché se pueden clasificar entre alguno de los siguientes:
  - *Compulsory miss*: la primera vez que se referencia un dato que no está en cache, produce un fallo de este tipo en caché, que da lugar a una transferencia de la línea de memoria principal a caché.

- *Capacity miss*: si la caché no puede contener todas las líneas que referencia durante la ejecución de un programa, aparecen fallos de "capacidad". Los nuevos datos que se referencien deben reemplazar a alguna de las líneas que había en caché (las líneas que se reemplazan dependen de la política de reemplazo elegida).
- *Conflict*: si la organización de la caché es directa o asociativa por conjuntos, los fallos por "conflicto" ocurrirán cuando el fallo provoque el reemplazo de una línea por otra sin que la caché esté completamente llena. Esto ocurre cuando demasiadas líneas que referencia el programa, se mapean en el mismo conjunto. A este tipo de fallo se le llama también fallo con "colisión" o fallo con "interferencia".

5. Ejemplo

Supongamos un sistema de memoria con palabras de 32 bits, y caches separadas de 64 palabras tanto para instrucciones como para datos. El tamaño del bloque será de 4 palabras. Queremos tener cuatro conjuntos en cada cache. Las políticas serán write back, write allocate, LRU y demand fetch. Debes cargar como fichero de traza de entrada el fichero `ex1.din` y como salida `xcache.out`.

`ex1.din`

```
2 0
0 0
2 4
0 40
2 8
0 80
2 c
0 c0
2 10
0 100
2 14
0 0
1 0
2 18
0 40
2 1c
0 80
2 20
0 c0
2 24
0 100
2 28
```

1. ¿Qué valores usarás para rellenar las casillas de la ventana de opciones de línea de comando que aparece al pulsar RUN DINERO en el menú SIMULATION?

   Asociatividad: ☐

   Tamaño de las caches: ☐

   Tamaño del bloque ☐

2. En el menú FILE, pulsa OPEN STREAM FILE y carga `ex1.din`. Si no están abiertas, abre las ventanas de memoria y cache. Justifica el tamaño de los campos en que se parte la dirección en las ventanas de cache. ¿Por qué la matriz de cajas que representa la cache es de 4x4?.

3. Pulsa el botón que tiene un triángulo verde hacia la derecha ("play") una vez. Analiza las ventanas de memoria y cache de instrucciones. ¿Por qué aparece una "R" en la primera palabra de memoria?. ¿A qué conjunto va el bloque pedido en la cache de instrucciones?. ¿De qué tipo es el fallo?

4. Si pulsas play otra vez, ¿a qué cache se accede ahora?.

5. Si pulsas play otra vez, se produce un acierto en la cache de instrucciones. ¿A qué palabra del bloque se accede? ¿Cómo se refleja en la ventana de memoria de instrucciones? ¿Qué crees que indica el número 2 que aparece sobre el bloque leído de la cache de instrucciones?. Pulsa dos veces sobre ese bloque y abre la ventana de traza ("Dinero Input") para tener más información.

6. En el menú OPTIONS pulsa MEMORY CONFIGURATION. Abre también la ventana de información de ciclos (icono con un reloj). Justifica por qué esta última ventana especifica 2 accesos de lectura, 8 palabras leídas y 10 ck necesarios para esas transferencias.

7. Ahora pulsa play 5 veces hasta que se llene el conjunto 0 de la cache de datos. El siguiente acceso es a la cache de instrucciones. El siguiente acceso a la cache de datos también es al conjunto 0. ¿Qué bloque sale de la cache?. ¿Cómo se refleja en la memoria de datos?

8. Pulsa play dos veces más (vamos por la línea 11 de la traza). ¿Qué tipo de fallo hay en la cache de datos? ¿Por qué?

9. En el siguiente acceso se produce una escritura. ¿Qué indica el color que ha tomado el bloque?. Recuerda simular la misma traza más tarde con política write through para ver qué color toma el bloque.

10. Pulsa play 6 veces más justificando los fallos que se producen. Ahora estamos en la línea 18 y el bloque que se cargó en la línea 12 está a punto de salir. Pulsa play dos veces más y explica qué ha pasado.

11. Termina la simulación. Abre ahora el fichero xcache.out (FILE-OPEN DINERO OUTPUT FILE). Justifica los valores que aparecen en la ventana de métricas.