Universidad Zaragoza
1542

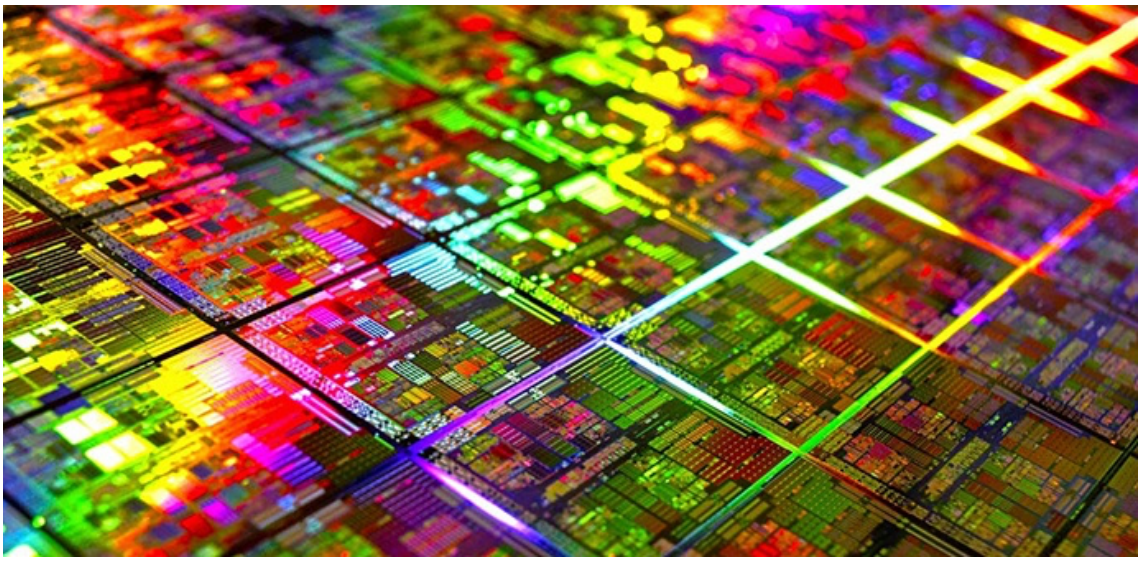
ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES 2

2º DE GRADO EN INGENIERÍA INFORMÁTICA, CURSO 2020/2021

UNIVERSIDAD DE ZARAGOZA

PRÁCTICA 6

ANÁLISIS DE ALTERNATIVAS EN EL DISEÑO DE MEMORIAS CACHE



Luis M. Ramos Javier Resano

José Luis Briz Francisco J. Martínez

Logisim
una herramienta de diseño y simulación de circuitos lógicos

El objetivo principal de esta práctica es comprender y manejar los distintos parámetros que definen el comportamiento de la memoria cache (tamaño de la cache, tamaño bloque, tipo de correspondencia, grado de asociatividad, política de escritura y reemplazo, etc.) y analizar las distintas alternativas de diseño del sistema de memoria, para obtener la mejor tasa de aciertos posible con unas determinadas restricciones físicas y económicas.

Para ello será necesario comprobar, utilizando un simulador, los diferentes parámetros de un sistema de cache para ver el efecto que tienen en el funcionamiento de la memoria. Estos parámetros son los siguientes:

- Tamaño de la Cache
- Cache unificada o dividida
- Tamaño del bloque en palabras
- Tipo de correspondencia
- Grado de asociatividad
- Política de reemplazo
- Política de escritura
- Número de niveles de cache

Simulador dineroIV

Podéis descargar el simulador de Mc (`dineroIV`) en la siguiente dirección:
http://pages.cs.wisc.edu/~markhill/DineroIV/

Este simulador acepta por la entrada estándar una traza en *formato dinero*[1], y genera por la salida estándar una estadística de tipos de acceso, fallos y tráfico de la Mc simulada.

Para el manejo del simulador podéis consultar el manual disponible. A pesar de que DineroIV no es un simulador de tiempo, también podéis intentar comentar cómo afectaría vuestro diseño al tiempo medio de acceso, pues aunque una cache de nivel 1 muy grande tendría una tasa de fallos muy pequeña, quizá el tiempo medio de acceso sería peor que una cache con un tamaño menor, aunque su tasa de fallos fuera ligeramente mayor.

Debéis pensar muy bien qué diferentes configuraciones queréis probar. Podéis crear un fichero .sh que contenga las diferentes configuraciones que queráis probar y los correspondientes ficheros de salida, para analizar los resultados obtenidos. De este modo podéis simular todas las combinaciones en muy poco tiempo con una sola orden. Luego podéis analizar tranquilamente los resultados.

En las **conclusiones** de vuestro informe incluso podéis añadir vuestra opinión sobre el simulador, indicando qué aspectos pensáis que se podrían mejorar.

[1] Formato documentado en el man de dineroIV en el que cada línea contiene:<*tipo*><*dirección*> donde *tipo* codifica si es acceso a instrucción, lectura de dato o escritura de dato, y *dirección* es la de la referencia a memoria.

Detalles de las trazas

Para ejecutar vuestro simulador debéis utilizar los archivos de traza que podéis obtener en la página de la asignatura en moodle. Para que los resultados sean más representativos, se recomienda el uso de todos los mismos (cc1, tex y spice) y de forma completa, en las diferentes pruebas.

Cada fichero tiene una línea por referencia, y a su vez cada línea contiene dos campos separados por un espacio en blanco: el primero indica el tipo de acceso a memoria, y el segundo campo indica la dirección de memoria (dirección de byte, no de bloque) expresada en hexadecimal. El tipo de acceso puede tener los siguientes valores:

- 0 --- Lectura de dato.
- 1 --- Escritura de dato.
- 2 --- Lectura de instrucción.

Detalles del simulador dinero IV

El simulador dinero es un simulador guiado por traza (trace-driven), es decir, el sistema recibe como entrada un fichero de trazas de memoria que contiene las referencias de memoria que pediría el procesador en un sistema real, durante la ejecución de un programa. El simulador leerá dicha secuencia de direcciones, las procesará y posteriormente devolverá los resultados obtenidos.

Dinero recibirá los parámetros desde la propia línea de comandos. Si ejecutamos la orden ./dineroIV –help, aparecerán todas las opciones disponibles. Aunque no vamos a utilizarlas todas, se recomienda mirarlas detenidamente (aparecen a continuación).

```
Usage: dineroIV [options]
Valid options:
 -help                Print this help message
 -copyright           Give details on copyright and lack of warranty
 -contact             Where to get the latest version or contact the authors
 -dineroIII           Explain replacements for Dinero III options
 -custom F            Generate and run custom simulator named F
 -lN-Tsize P          Size
 -lN-Tbsize P         Block size
 -lN-Tsbsize P        Sub-block size (default same as block size)
 -lN-Tassoc U         Associativity (default 1)
 -lN-Trepl C          Replacement policy
                       (l=LRU, f=FIFO, r=random) (default l)
 -lN-Tfetch C         Fetch policy
                (d=demand, a=always, m=miss, t=tagged,
                l=load forward, s=subblock) (default d)
```

```
-lN-Tpfdist U          Prefetch distance (in sub-blocks) (default 1)
-lN-Tpfabort U         Prefetch abort percentage (0-100) (default 0)
-lN-Twalloc C          Write allocate policy
                (a=always, n=never, f=nofetch) (default a)
-lN-Twback C           Write back policy
                (a=always, n=never, f=nofetch) (default a)
-lN-Tccc               Compulsory/Capacity/Conflict miss statistics
-skipcount U           Skip initial U references
-flushcount U          Flush cache every U references
-maxcount U            Stop simulation after U references
-stat-interval U       Show statistics after every U references
-informat C            Input trace format
                (D=extended din, d=traditional din, p=pixie32, P=pixie64,
                b=binary) (default D)
-on-trigger A          Trigger address to start simulation
-off-trigger A         Trigger address to stop simulation
-stat-idcombine        Combine I&D cache stats

Key:
U   unsigned decimal integer
S   like U but with optional [kKmMgG] scaling suffix
P   like S but must be a power of 2
C   single character
A   hexadecimal address
F   string
N   cache level (1 <= N <= 5)
T   cache type (u=unified, i=instruction, d=data)
```

Para utilizar el simulador dineroIV previamente tendréis que compilarlo (se recomienda hacerlo en linux, para ello mirad el fichero README).

En la orden de invocación de `dineroIV` se pueden especificar numerosas opciones (dimensionado, políticas de escritura ...). A continuación, se muestra un ejemplo de llamada al simulador realizada de forma correcta:

```
zcat tex.din.Z | dineroIV -informat d -l1-isize 16K -l1-dsize 8K -l1-ibsize 32 -l1-dbsize 32
```

que simula una Mc separada de 16KB para instrucciones y 8KB para datos de correspondencia directa (es el valor por defecto) con tamaño de bloque de 32B.

En el caso que descomprimáis las trazas, también se puede ejecutar de la siguiente forma:

```
./dineroIV -l1-usize 64k -l1-ubsize 32 -l1-uassoc 2 -l1-urepl l -informat d < tex.trace
> resultados.txt

./dineroIV -l1-isize 32k -l1-ibsize 32 -l1-dsize 32k -l1-dbsize 32 -l1-dassoc 2 -l1-
drepl l -informat d < gcc.trace >> resultados.txt
```

Las trazas contienen alrededor de un millón de referencias y están comprimidas (`.z`) para ahorrar espacio de disco.

Informe a entregar

1. Se pide **realizar un informe** con las variaciones en el rendimiento de la cache en función de la traza utilizada, de la política de escritura, del tamaño de la cache, del tamaño de bloque, el número de referencias procesadas, el grado de asociatividad, etc. Realizad los **experimentos variando los diferentes parámetros del diseño.**

   Debéis tener en cuenta que para que los resultados sean más representativos, sería necesario usar las trazas de diferentes programas (p.ej., cc1, tex y spice).

2. Debéis completar el informe con las figuras y tablas que consideréis necesarias para que se entienda bien vuestro trabajo.

3. Una vez hayáis hecho el análisis, deberéis presentar una propuesta de diseño de cache que obtenga la menor tasa de fallos posible, suponiendo que **únicamente podemos utilizar físicamente** un determinado número de bits (**2300 KB**), para implementar tanto la RAM de datos como la RAM de etiquetas (e información de control).