

Ingeniería en Informática 2º curso



ESTRUCTURA DE COMPUTADORES

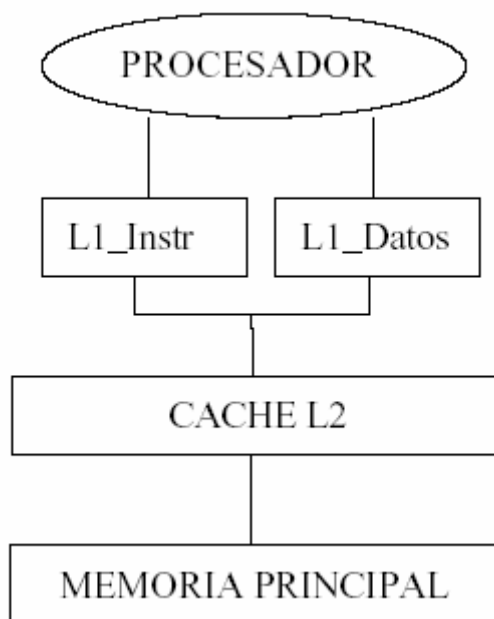
Práctica 3: Cache

David Jesús Horat Flotats

Objetivos:

El principal objetivo de esta práctica es el afianzamiento de los conceptos de la jerarquía de memoria y su evolución al modificar los diferentes parámetros. Para ello tendremos que aprender a utilizar la herramienta de simulación “sim-cache” y con la batería de pruebas SPEC, que es el estándar para las pruebas de rendimiento.

El profesor nos repartirá un programa de enteros y uno de punto flotante correspondiente a los SPEC (en mi caso han sido el GCC y el MGRID) y con ellos haremos cinco tandas de pruebas para comprobar las evoluciones de la cache de primer nivel dependiendo de ciertos parámetros que corresponden a las actividades de la 3 a la 7 del dossier de la práctica 3. Al final de esta memoria tendremos un anexo que nos resuelva las preguntas de la actividad 1.



A la izquierda vemos un esquema de la jerarquía de memoria. Nosotros nos centraremos en la L1 tanto de instrucciones como de datos e incluso cuando ambas estén unidas en una sola. La L2 se dejará fija siendo una caché unificada con los parámetros: 4096:64:2:1 que corresponden al número de bloques, el tamaño de bloque, la asociatividad y la política de reemplazo, que en este caso será LRU (Less Recently Used). Una vez aclarados estos puntos, vamos a ver los pasos previos a seguir.

Pasos previos:

Teniendo en cuenta la gran cantidad de pruebas a realizar, he decidido automatizar estas usando archivos de procesos por lotes (.bat) creando una jerarquía de 4 niveles para modular el código y evitar repeticiones. A continuación explicaré cada uno de los niveles:

Nivel 1: Corresponde al archivo “p3.bat” que es el que se ejecutará y será el que llame a cada uno de los archivos del siguiente nivel. Su código, el cual es muy sencillo, es:

```
cls
echo Práctica 3 de Estructura de Computadores: Cache
call p3a3.bat
call p3a4.bat
call p3a5.bat
call p3a6.bat
call p3a7.bat
echo Fin práctica 3
```

Nivel 2: Los archivos del nivel 2, que como vemos en el código del nivel anterior son: p3a3.bat, p3a4.bat, p3a5.bat, p3a6.bat y p3a7.bat y se corresponden con cada una de las actividades que son la base de esta memoria. Su código lo veremos en profundidad en cada una de las secciones correspondientes de las actividades.

Nivel 3: Debido a que hemos de realizar tandas de pruebas para 2 programas SPEC (uno de enteros y otro de flotantes) decidí crear archivos “.bat” para cada cada uno de los programas SPEC y para cada práctica, pasando como parámetros únicamente los que varíasen en dicha actividad. La nomenclatura usada en este caso es:

act[numero][programa].bat. Un ejemplo puede ser: act3i.bat que corresponde al programa de la actividad 3 del programa correspondiente a enteros (integer). Cada uno de estos archivos lo analizaremos en su correspondiente actividad.

Nivel 4: Una vez realizada cada una de las pruebas, deberemos extraer únicamente los resultados que queramos, ya que probablemente la mayoría de los demás sean exactamente iguales. Hemos de recordar que solo cambiamos los parámetros de una de las cache por lo que sólo nos interesa el dato que vayamos a estudiar de esa determinada

cache. A lo largo de esta práctica veremos como sólo necesitamos 4 grupos de datos: tasa de fallos de la caché de instrucciones de nivel 1, tasa de fallos de la cache de datos de nivel 1, tasa de fallos tanto de la caché de instrucciones de nivel 1 como la de la caché de datos de nivel 1 y la tasa de fallos de la cache de nivel 1 unificada. Para cada uno de estos 4 grupos hemos creado un archivo que nos extraerá y almacenará en un archivo (pasado por parámetros) los datos concretos. El código es el siguiente:

“il1Stats.bat”: Tasa de fallos de la cache de instrucciones de nivel 1

```
find "il1.miss_rate" < prueba1.txt >> %1  
echo _____ >> %1
```

“dl1Stats.bat”: Tasa de fallos de la caché de datos de nivel 1

```
find "dl1.miss_rate" < prueba1.txt >> %1  
echo _____ >> %1
```

“idl1Stats.bat”: Tasa de fallos de las cache de datos e instrucciones de nivel 1

```
find "il1.hits" < prueba1.txt >> %1  
find "il1.misses" < prueba1.txt >> %1  
find "dl1.hits" < prueba1.txt >> %1  
find "dl1.misses" < prueba1.txt >> %1  
echo _____ >> %1
```

“ul1Stats.bat”: Tasa de fallos de la cache de nivel 1 unificada

```
find "ul1.miss_rate" < prueba1.txt >> %1  
echo _____ >> %1
```

Una vez vistos los archivos comunes a todas las prácticas, vamos ya a analizar cada una de las actividades.

ACTIVIDAD 3: Estudio de la tasa de fallos de la cache de instrucciones de nivel 1 en función del tamaño de bloque y la asociatividad

En esta primera actividad estudiaremos el comportamiento de la cache de instrucciones de nivel 1 al modificar el tamaño de los bloques a 8, 16, 32 y 64 bytes y para asociatividades 1, 2, 4, 8 y totalmente asociativa teniendo en cuenta que será de 32 Kbytes y con una política de reemplazo LRU. La configuración estándar para estas pruebas será:

Cache	Tamaño total	Tamaño bloque	Asociatividad	Reemplazo
L1_Datos	32 Kbytes	64 bytes	2	LRU
L2	512 Kbytes	64 bytes	2	LRU

Dado que los parámetros a modificar son únicamente los de tamaño de bloque, asociatividad y por tanto el número de bloques, estos serán los únicos que pasaremos por parámetro a nuestros archivos “act3i.bat” y “act3f.bat” que a continuación listamos:

“act3i.bat”

```
echo i %1 >> p3a3.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 il1:%1:1 -
cache:dl1 dl1:256:64:2:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 gcc.ss amptjp.i
call il1Stats.bat p3a3.txt
```

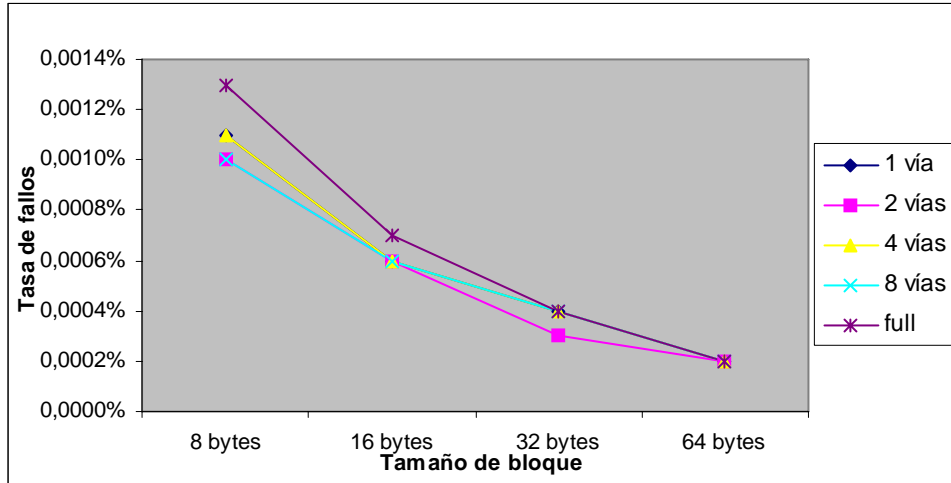
“act3f.bat”

```
echo f %1 >> p3a3.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 il1:%1:1 -cache:dl1
dl1:256:64:2:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 mgrid.ss<mgrid.in
call il1Stats.bat p3a3.txt
```

Una vez estos listos, creamos el archivo “p3a3.bat” que llamará a cada uno de los archivos listados anteriormente para cada una de las configuraciones posibles antes descritas. Con ello generará un archivo de resultados llamado “p3a3.txt”. Nosotros presentaremos los resultados en forma de tabla y gráficas para cada uno de los dos programas (entero y punto flotante) que es como nos pide el enunciado y a continuación expondremos nuestras conclusiones.

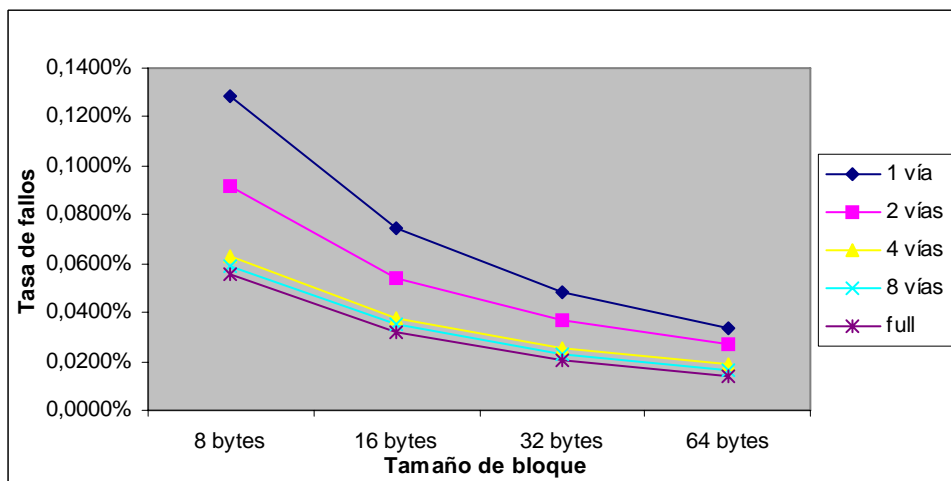
GCC

	1	2	4	8	Full
8 bytes	0.0011	0.0010	0.0011	0.0010	0.0013
16 bytes	0.0006	0.0006	0.0006	0.0006	0.0007
32 bytes	0.0004	0.0003	0.0004	0.0004	0.0004
64 bytes	0.0002	0.0002	0.0002	0.0002	0.0002



MGRID

	1	2	4	8	Full
8 bytes	0.1282	0.0913	0.0631	0.0590	0.0554
16 bytes	0.0749	0.0544	0.0378	0.0350	0.0321
32 bytes	0.0485	0.0369	0.0254	0.0230	0.0204
64 bytes	0.0334	0.0273	0.0185	0.0167	0.0138



ACTIVIDAD 4: Estudio de la tasa de fallos de la caché de datos de nivel 1 en función del tamaño de bloque y la asociatividad

Esta actividad unicamente varía con respecto a la anterior en que en vez de la cache de instrucciones, se estudiará la cache de datos. Las configuraciones posibles son las mismas que las anterior, es decir, que haremos pruebas para un cache de datos de 32 Kbytes y política de reemplazo LRU, cuyo tamaño de bloque será de 8, 16, 32 o 64 bytes y con asociatividad 1, 2, 4, 8 o totalmente asociativo. Las configuraciones de las demás cache quedarán como se refleja en la siguiente tabla:

Cache	Tamaño total	Tamaño bloque	Asociatividad	Reemplazo
L1_Instr	32 Kbytes	64 bytes	2	LRU
L2	512 Kbytes	64 bytes	2	LRU

Seguidamente crearemos los archivos “act4i.bat” y “act4f.bat” que serán incluirán los parámetros estáticos que hemos impuestos y a los que se les pasará por parámetros los que irán variando. Su código es el siguiente:

“act4i.bat”

```
echo i %1 >> p3a4.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 il1:256:64:2:1
-cache:dl1 dl1:%1:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 gcc.ss amptjp.i
call dl1Stats.bat p3a4.txt
```

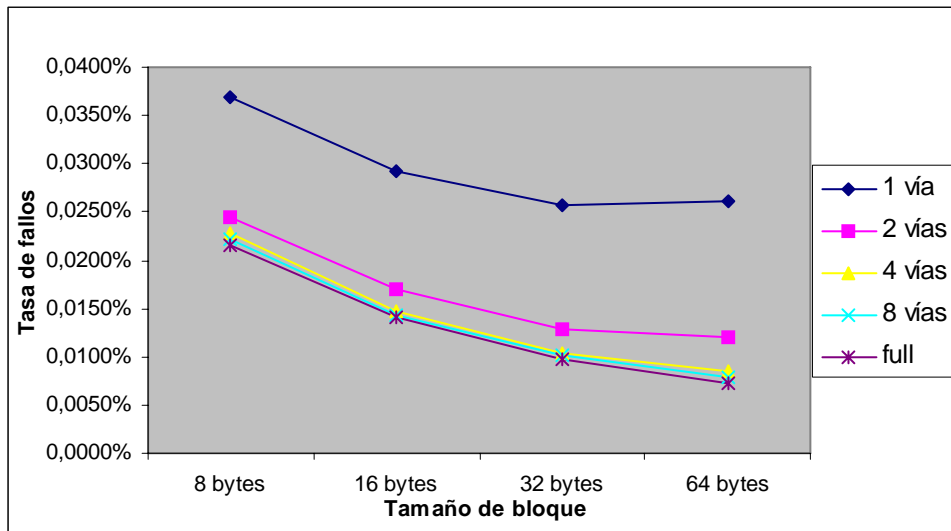
“act4f.bat”

```
echo f %1 >> p3a4.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 il1:256:64:2:1 -
-cache:dl1 dl1:%1:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 mgrid.ss<mgrid.in
call dl1Stats.bat p3a4.txt
```

A continuación creemos el archivo “p3a4.bat” que será el encargado de ejecutar los archivos anteriores con los parámetros correspondientes para todas las configuraciones posibles. Una vez obtenido el archivo “p3a4.txt” en donde obtenemos los resultados, el paso siguiente es crear las tablas de resultados, las gráficas y extraer conclusiones.

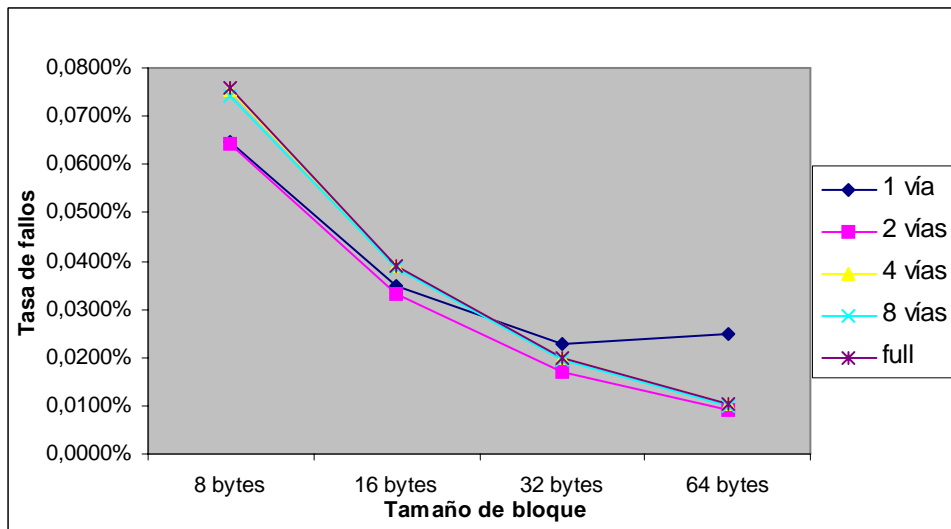
GCC

	1	2	4	8	Full
8 bytes	0.0369	0.0245	0.0227	0.0222	0.0215
16 bytes	0.0293	0.0169	0.0147	0.0143	0.0140
32 bytes	0.0256	0.0129	0.0104	0.0101	0.0097
64 bytes	0.0262	0.0120	0.0085	0.0079	0.0073



MGRID

	1	2	4	8	Full
8 bytes	0.0648	0.0644	0.0756	0.0742	0.0759
16 bytes	0.0349	0.0333	0.0390	0.0384	0.0388
32 bytes	0.0228	0.0171	0.0198	0.0196	0.0197
64 bytes	0.0247	0.0092	0.0103	0.0101	0.0102



ACTIVIDAD 5: Estudio de la tasa de fallos de la caché de instrucciones de nivel 1 en función de la asociatividad y del tamaño de la cache

En esta actividad volveremos a estudiar el comportamiento de la cache de primere nivel pero en este caso los parámetros a variar son la asociativad entre 1, 2, 4, 8 o totalmente asociativa y los tamaños de cache entre 1, 4, 16, 32, 64 y 128 KBytes. Sin embargo lo que no variará será el tamaño de bloque que se mantendrá en 64 bytes y la política de reemplazo será siempre LRU (Less Recently Used). Las demás demás cache quedarán configuradas como se especifica en la siguiente tabla:

Cache	Tamaño total	Tamaño bloque	Asociatividad	Reemplazo
L1_Data	32 Kbytes	64 bytes	2	LRU
L2	512 Kbytes	64 bytes	2	LRU

A continuación listamos los archivos que deberemos haber creado “act5i.bat” y “act5f.bat” que incluirán lo mismo que los anteriores pero específicamente para esta actividad:

“act5i.bat”

```
echo i %1 >> p3a5.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 il1:%1:l -cache:dl1
dl1:256:64:2:l -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:l gcc.ss amptjp.i
call il1Stats.bat p3a5.txt
```

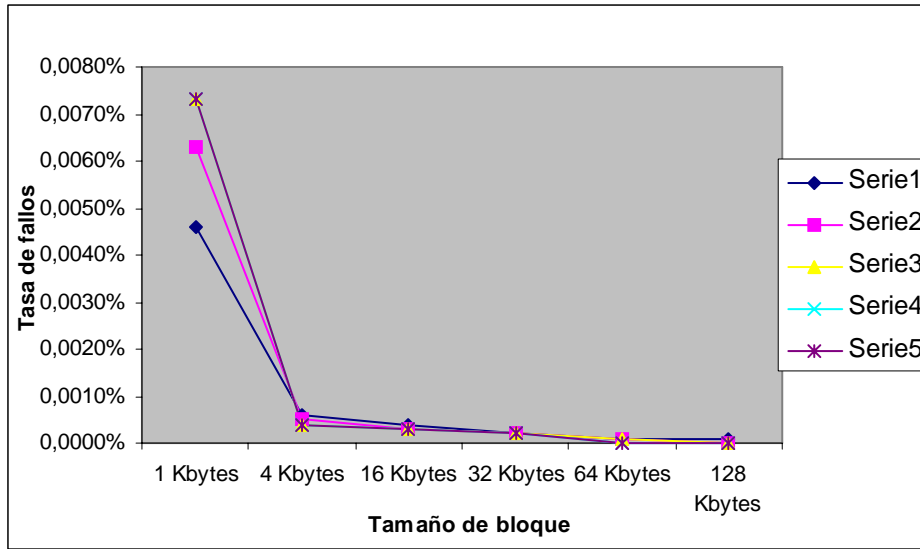
“act5f.bat”

```
echo f %1 >> p3a5.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 il1:%1:l -cache:dl1
dl1:256:64:2:l -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:l mgrid.ss<mgrid.in
call il1Stats.bat p3a5.txt
```

Ahora hacemos el archivo “p3a5.bat” configurado para dar los resultados de las configuraciones posibles que se obtendrán en “p3a5.txt”. Una vez obtenidos los resultados hacemos las tablas, los gráficos y las conclusiones.

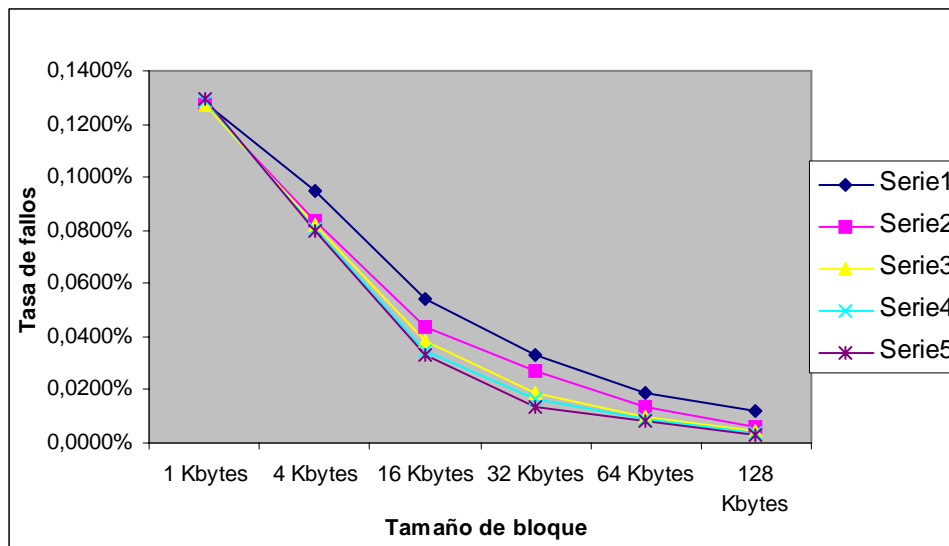
GCC

	1	2	4	8	Full
1 Kbytes	0.0046	0.0063	0.0073	0.0073	0.0073
4 Kbytes	0.0006	0.0005	0.0004	0.0004	0.0004
16 Kbytes	0.0004	0.0003	0.0003	0.0003	0.0003
32 Kbytes	0.0002	0.0002	0.0002	0.0002	0.0002
64 Kbytes	0.0001	0.0001	0.0001	0.0000	0.0000
128 Kbytes	0.0001	0.0000	0.0000	0.0000	0.0000



MGRID

	1	2	4	8	Full
1 Kbytes	0.1279	0.1274	0.1275	0.1285	0.1294
4 Kbytes	0.0948	0.0834	0.0822	0.0802	0.0801
16 Kbytes	0.0540	0.0434	0.0385	0.0344	0.0331
32 Kbytes	0.0334	0.0273	0.0185	0.0167	0.0138
64 Kbytes	0.0186	0.0139	0.0101	0.0089	0.0085
128 Kbytes	0.0120	0.0059	0.0043	0.0036	0.0028



ACTIVIDAD 6: Análisis de la influencia de la política de reemplazo en la tasa de fallos de la cache de instrucciones de nivel 1

A continuación estudiaremos como afecta el tipo de política de reemplazo en las cache de instrucciones dependiendo de su tamaño. Para ello configuraremos dicha cache con un tamaño de bloque de 64 bytes y con una asociatividad de 4, por lo que tendremos que adaptar el número de conjuntos para que se ajusten a los tamaños de 1, 4, 16, 32, 64 y 128 Kbytes. Examinaremos 3 tipos de políticas de reemplazo:

- LRU (Less Recently Used): Donde se reemplaza el bloque menos usado
- FIFO (First In First Out): Donde se reemplaza el bloque que entro el primero
- RANDOM: Donde se reemplaza un bloque aleatoriamente

Las configuraciones de las demás cache vienen dadas por la siguiente tabla:

Cache	Tamaño total	Tamaño bloque	Asociatividad	Reemplazo
L1_Data	32 Kbytes	64 bytes	2	LRU
L2	512 Kbytes	64 bytes	2	LRU

Como ya es habitual, creamos los ficheros “act6i.bat” y “act6f.bat” que listamos a continuación:

“act6i.bat”

```
echo i %1 >> p3a6.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 il1:%1:64:4:%2 -
cache:d11 d11:256:64:2:1 -cache:il2 dl2 -cache:d12 ul2:4096:64:2:1 gcc.ss amptjp.i
call il1Stats.bat p3a6.txt
```

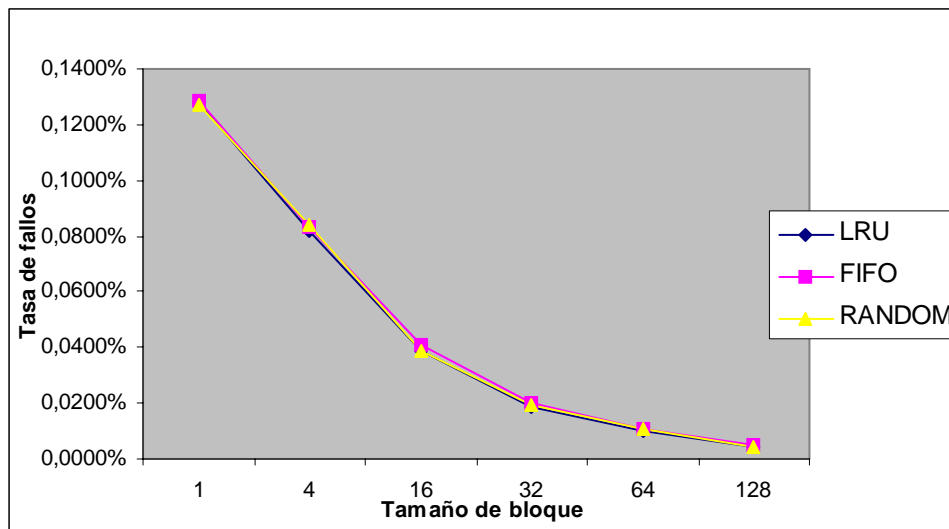
“act6f.bat”

```
echo f % 1:%2 >> p3a6.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 il1:% 1:64:4:%2 -
cache:dl1 dl1:256:64:2:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 mgrid.ss<mgrid.in
call il1Stats.bat p3a6.txt
```

Seguidamente hacemos el archivo “p3a6.bat” que una vez ejecutado creará el archivo de resultados “p3a6.txt”. Ahora mostraremos dichos resultados en forma de tabla, gráfica y conclusión.

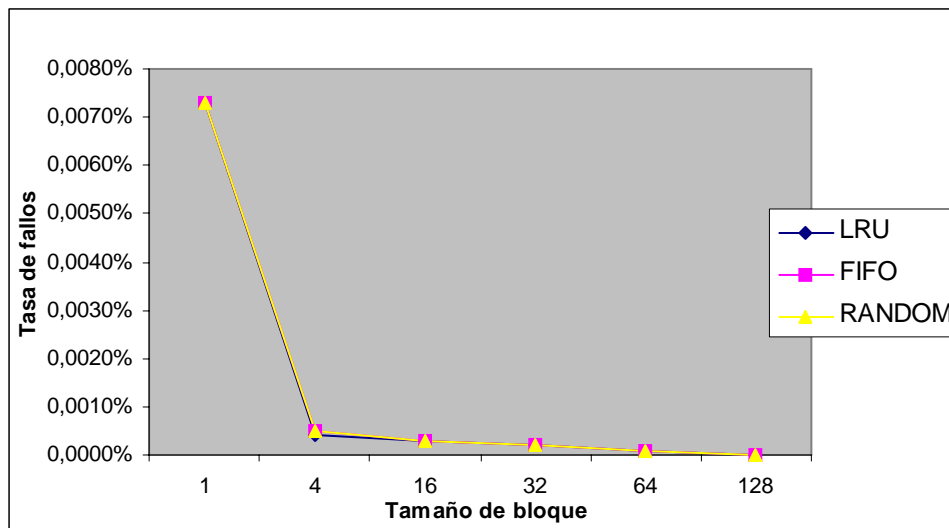
GCC

	1	4	16	32	64	128
LRU	0.1275	0.0822	0.0385	0.0185	0.0101	0.0043
FIFO	0.1286	0.0834	0.0408	0.0198	0.0110	0.0048
RANDOM	0.1271	0.0841	0.0389	0.0195	0.0108	0.0046



MGRID

	1	4	16	32	64	128
LRU	0.0073	0.0004	0.0003	0.0002	0.0001	0.0000
FIFO	0.0073	0.0005	0.0003	0.0002	0.0001	0.0000
RANDOM	0.0073	0.0005	0.0003	0.0002	0.0001	0.0000



ACTIVIDAD 7: Estudio de la tasa de fallos de las caches de nivel 1 en función del tamaño y de tener las caches de nivel 1 unificadas o separadas

Por último observaremos la cache de nivel 1 en conjunto, diferenciando entre caches unidas o separadas en instrucciones y datos y atendiendo al tamaño. Como criterio tenemos que una cache unificada de X Kbytes se compara con dos cachés separadas de X/2 Kbytes para instrucciones y datos. En este caso los datos estáticos para la cache de nivel 1 son que el tamaño de bloque será de 64 bytes, la asociatividad será 2 y la política de reemplazo será LRU. Por otro lado se harán pruebas con las caches tanto separadas como unificadas y para tamaños de 32, 64 y 128 Kbytes. La configuración de la cache de nivel 2 viene dada por la siguiente tabla:

Cache	Tamaño total	Tamaño bloque	Asociatividad	Reemplazo
L2	512 Kbytes	64 bytes	2	LRU

Una vez conocidos todos los datos iniciales vemos que ya no es suficiente con 2 archivos a los cuales les pasemos por parámetros las configuraciones variables, sino que al ser baterías de pruebas para 2 programas (GCC y MGRID) y teniendo en cuenta que es para caches unificadas y separadas tendremos que crear 4 archivos:

“act7fs.bat”

```
echo fs %1 >> p3a7.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 il1:%1:64:2:1 -
cache:dl1 dl1:%1:64:2:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 mgrid.ss<mgrid.in
call idl1Stats.bat p3a7.txt
```

“act7fu.bat”

```
echo fu %1 >> p3a7.txt
sim-cache -max:inst 90000000 -redir:sim prueba1.txt -cache:il1 dl1 -cache:dl1
ul1:%1:64:2:1 -cache:il2 none -cache:dl2 none mgrid.ss<mgrid.in
call ul1Stats.bat p3a7.txt
```

“act7is.bat”

```
echo is %1 >> p3a7.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 il1:%1:64:2:1 -
cache:dl1 dl1:%1:64:2:1 -cache:il2 dl2 -cache:dl2 ul2:4096:64:2:1 gcc.ss amptjp.i
call idl1Stats.bat p3a7.txt
```

“act7iu.bat”

```
echo iu %1 >> p3a7.txt
sim-cache -max:inst 50000000 -redir:sim prueba1.txt -cache:il1 dl1 -cache:dl1
ul1:%1:64:2:1 -cache:il2 none -cache:dl2 none gcc.ss amptjp.i
call ul1Stats.bat p3a7.txt
```

Ya para finalizar esta actividad, y una vez creado el correspondiente archivo “p3a7.bat”, expondremos los resultados en forma de tablas, gráficas y conclusiones.

GCC

	32	64	128
Unificadas	0.0330	0.0179	0.0086
Separadas (inst+dat)	0.0038	0.0024	0.0012

MGRID

	32	64	128
Unificadas	0.0043	0.0031	0.0028
Separadas (inst+dat)	0.0046	0.0037	0.0027

Debido a la simplicidad de las tablas, hemos decidido no realizar gráficas pero si hacerle un comentario especial. He tenido problemas al usar una caché unificada L1 con una caché L2, por lo que en ambas pruebas, en la parte de caché unificadas, he quitado la caché L2, pero en las separadas si la he puesto con las configuraciones explicadas anteriormente. Posiblemente por este motivo la tasa de errores de la caché unificada se dispara en el GCC ya que requiere mucha memoria como pudimos comprobar por su extenso tamaño. Sin embargo el MGRID al ser un programa pequeño no se ha visto

afectado por este problema, es más, al usarse mucho la caché de datos, se ha beneficiado de que estuvieran juntas y así poder más memoria para datos. Por lo cual podemos concluir que claramente la cache depende de los programas a usar, sin embargo en estas comprobaciones nos quedaríamos con una caché separada ya que al hacer ponderación global, vemos que por término medio será más beneficiosa.

Conclusiones:

En la actividad 3 estudiamos la tasa de fallos de la cache de instrucciones de nivel 1 en función del tamaño de bloque y la asociatividad. Como podemos comprobar en las gráficas, tanto para el programa de enteros (GCC) como para el de coma flotante (MGRID) la tasa de fallos disminuye cuanto más grande es el tamaño de bloque y cuanto más asociativa es. Era un resultado de esperar ya que si capturamos más datos, menos capturas deberemos de hacer y si tenemos una asociatividad mayor, podremos situar más datos de una sección en la cache. Sin embargo vemos que al ser el tamaño máximo de bloque 64 bytes y el tamaño de la cache es de 32 Kbytes vemos como no existe punto de polución aun por lo cual se podría seguir aumentando. Algo a destacar es quizás que cuanto mayor es el tamaño de bloque más se reduce la diferencia entre distintas asociatividades.

En la actividad 4 estudiamos la tasa de fallos de la cache de datos de nivel 1 en función del tamaño de bloque y la asociatividad. Los datos son parecidos al de la actividad anterior, lógicamente. Lo único que destacaremos será la gran diferencia de la asociatividad 1 con respecto a las otras, vemos que se hace uso intensivo de ciertas secciones de la memoria y por eso se requiere una asociatividad mayor para obtener un rendimiento óptimo.

En la actividad 5 estudiamos la tasa de fallos de la cache de instrucciones de nivel 1 en función de la asociatividad y del tamaño de la cache. Vemos claramente como para una cache pequeña se dispara la tasa de fallos y tenemos que llegar hasta los 4 Kbytes para tener un rendimiento aceptable, al menos en el programa de enteros. En el programa de coma flotante se nota menos ya que hay tasas de fallos muy grandes y van casi de forma gradual.

En la actividad 6 hicimos un análisis de la influencia de la política de reemplazo en la tasa de fallos de la cache de instrucciones de nivel 1. Vemos como la política de reemplazo influye minimamente aunque cuanto mayor es el tamaño mejor se comporta la política LRU (Less Recently Used) en valores relativos en comparación con las demás.

Anexo I: Respuestas a las preguntas de la 1º actividad de la práctica 3

1. ¿Qué porcentaje de las instrucciones ejecutadas son referencias a memoria?

Viene dado por `sim_num_refs` y da el valor de 25481190.

2. ¿Por qué `il1.writebacks` es igual a cero?

Porque las instrucciones no se escriben.

3. ¿Por qué `il1.invalidations` es igual a cero?

Porque no se ha escrito en memoria antes que en la cache.

4. ¿Cuál es la `il1.hit_rate`? ¿De qué maneras puede calcularse?

Se puede calcular bien haciendo la división `hits/accesses` (aciertos/accesos), o bien haciendo la resta `1-miss_rate` (1-tasa de fallos). En ambos casos, el resultado es: 0,9755

5. ¿Qué organización tiene la cache de nivel L2? ¿Cuáles serían los parámetros de diseño para hacerla totalmente asociativa?

En este caso usa la configuración por defecto, que es una cache de nivel unificada con 1024 conjuntos, 64 bytes de tamaño de bloque y una asociatividad 4.

Para configurar esta cache como totalmente asociativa y cogiendo 64 bytes de tamaño de bloque, tan solo tendríamos que despejar de la ecuación:

$1 * 64 * X = 256 * 1024$ cuyo resultado es $X = 4096$ y por lo tanto usaríamos una asociatividad de 4096 y los parámetros del sim-cache quedarían: 1:64:4096.

6. ¿Qué significado tiene ul2-accesses? Su valor en el ejemplo es de 4.790.064 ¿Cómo se obtiene este valor a partir de los otros datos que nos suministra el simulador?

Representan los accesos a la cache de nivel 2, que son el número de fallos que haya tenido la cache de nivel 1. Se puede calcular también como $ul2.hit + ul2.misses$

7. Atendiendo a los datos de salida de la simulación, ¿qué caché crees que tiene un mejor comportamiento, la il1 o la dl1? ¿Por qué se comporta mejor?

La cache de instrucciones de nivel 1 se comporta mejor que la cache de datos de nivel 1 como se puede observar al comparar la tasa de fallos ($miss_rate$) de ambas. Ello es debido a que el acceso a las instrucciones es más secuencial que el acceso a datos, ya que la mayoría de los programas suelen estar bien estructurados y los datos pueden ser salteados, como por ejemplo las matrices.