UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

ORGANIZACIÓN DE COMPUTADORES

LABORATORIO 3

Profesores: Alfonso Guzmán & Daniel Wladdimiro
Ayudantes: Pablo Ulloa & Ariel Undurraga

Santiago - Chile
10 de mayo de 2017

CAPÍTULO 1. CONTEXTO

Como lo confirma la teoría vista en cátedra, uno de los factores que inciden en el rendimiento de un programa es la cantidad de accesos a memoria que debe realizar el procesador para obtener datos en tiempo de ejecución. Para evitar el buscar datos en la memoria principal. Se utilizan memorias de tipo cache, las cuales son de un tamaño reducido (en comparación a la memoria principal) y se ubican de forma cercana al procesador, por esta misma razón es que permiten una obtención de datos considerablemente más acelerada que la memoria principal. Sin embargo, las memorias cache, por ser limitadas, no pueden contener todos los datos existentes en la memoria principal, por lo que estos datos deben ser ingresados por bloques a una dirección determinada de la cache, y en caso que dicha dirección esté ocupada, se debe reemplazar el contenido anterior por el que está ingresando a la cache.

Para identificar la ubicación del bloque en caché existe un índice, el cual dependiendo de la configuración del caché (si es mapeo directo, full asociativo o de n-vías), determinará la posición en la que el dato consultado se buscará. En caso de que sea necesario aplicar un reemplazo, ya que en el índice consultado no se puedan ingresar más datos, entran en acción las políticas de reemplazo que el caché en cuestión utiliza; entre estas políticas se encuentran, por ejemplo: MRU, LRU, LIFO y FIFO.

Cabe preguntarse según lo anterior entonces, ¿Cómo saber qué configuración de caché utilizar?, ¿Cuál me entregará una mayor tasa de hit?, ¿Podría darse que configuraciones muy distintas se lleguen a comportar de forma similar?.

CAPÍTULO 2. INSTRUCCIONES

En el presente laboratorio se propone desarrollar un programa que simule el comportamiento de la caché con sólo un nivel, cuyo tamaño de palabra corresponde a 32 bits. Para esto, es necesario leer un archivo de entrada, donde estén los datos que se consultarán a la caché, los cuales serán representados por números enteros separados por un salto de línea.

Para la configuración del laboratorio es necesario ingresar los parámetros utilizando la función `getopt` de la librería `unistd.h`. De esta manera, los parámetros que deben establecerse son:

Parámetro	Bandera	Valores	Descripción
Política de reemplazo	-r	LRU/MRU/FIFO	Se refiere a la política de reemplazo que se va a aplicar en el caché, el cual puede ser FIFO, LRU o MRU.
n-vías	-v	[1-n]	Se refiere al grado de asociatividad de la caché. Donde $n \leq q$ y $n = 2^x; x \in N$
Cantidad de palabras	-p	[1-p]	Se refiere a la cantidad de palabras por bloque que se posee en caché, el cual define el tamaño del bloque. Donde $p = 2^x; x \in N$
Cantidad de bloques	-t	[1-q]	Se refiere a la cantidad de bloques que posee el caché, lo cual define el tamaño de éste. Donde $q = 2^x; x \in N$

Supongamos que se desea configurar un caché con mapeo directo con bloques de 2 palabras, de tamaño 4KB y política de reemplazo LRU. De ser así, entonces la ejecución del programa sería de la siguiente forma:

```
./nombreLaboratorio3 -r LRU -v 1 -p 2 -t 512
```

Vale decir que esta librería se encuentra solamente disponible para sistemas operativos Unix, por lo que en caso de utilizar Windows, la entrada deberá seguir la misma estructura sin embargo sin las flags, teniendo que obtener estos datos directamente a partir del arreglo argv.

```
nombreLaboratorio3.exe LRU 1 2 512
```

Debe especificarse dentro del informe el sistema operativo a ocupar para la revisión. En caso de no ser así, se asumirá que el programa fue implementado con `getopt` y se evaluará utilizando ésta librería.

Por otra parte, verifiquemos un ejemplo de cómo debería ser el funcionamiento de la caché. Analicemos una caché que posee 16 bloques, con 1 palabra por bloque y de mapeo directo. De ser así, al inicializar la caché obtendremos lo siguiente:

i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

En el caso que tengamos una caché que posee 8 bloques, con 2 palabras por bloque y de mapeo directo, tendríamos lo siguiente:

i_0		i_1		i_2		i_3		i_4		i_5		i_6		i_7	
p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Por otra parte, en el caso que poseamos una caché que posee 8 bloques, con 1 palabra por bloque y de 2 vías asociativo, tendríamos lo siguiente:

i_0		i_1		i_2		i_3		i_4		i_5		i_6		i_7	
v_0	v_1	v_0	v_1	v_0	v_1	v_0	v_1	v_0	v_1	v_0	v_1	v_0	v_1	v_0	v_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Finalmente, se deben crear dos archivos de salida, dónde el primero corresponde a los datos alojados en la caché posterior a la última consulta realizada a éste. Y el segundo, las estadísticas de la caché, considerando la tasa de miss y la tasa de hit.

Para la inserción y consulta de los datos, se debe aplicar el procedimiento visto en la ***cátedra***.

2.1 EXIGENCIAS

- El programa debe estar escrito en C o C++ (estándar ANSI C). En caso de usar C++ se exigirá orientación a objetos, de caso contrario no será revisado.
- El programa y el informe deben ser entregados como una carpeta comprimida en formato `zip` o `tar.gz`-
- El programa debe ser entregado en su código fuente junto a un archivo Makefile para su compilación.
- El programa debe tener usabilidad. El usuario debe ser capaz de ingresar el nombre del archivo a leer y los nombres de los archivos de salida por una interfaz.

- El programa debe cumplir con un mínimo de calidad de software (funciones separadas y nombres tanto de funciones como de variables, de fácil comprensión).
- El informe escrito no debe exceder 10 páginas de texto escrito, sin considerar portada ni índice, en caso contrario, por cada página extra, se descontará 5 décimas.
- El informe escrito debe ser entregado en formato PDF, por lo que puede ser desarrollado en LaTeX, Microsoft Word, OpenOffice Writer, etc.

2.2 RECOMENDACIONES

- Utilizar la plantilla de LaTeX disponible en el Moodle del curso para la elaboración del informe.
- Consultar a los ayudantes y en Moodle del curso.

2.3 DESCUENTOS

- Por cada exigencia no cumplida, se descontarán dos décimas a la nota, a excepción las que ya mencionan su descuento.
- Por cada día de atraso, se descontará un punto a la nota.
- Por cada tres faltas ortográficas o gramaticales en el informe, se descontará una décima a la nota.
- Por cada falta de formato en el informe, se descontará una décima a la nota.

2.4 EVALUACIÓN

- La nota del laboratorio será el promedio aritmético del código fuente con el informe, y en caso de obtener una nota inferior a 4 en alguno de las dos calificaciones, se evaluará con la menor nota.
- En caso que no se entregue alguno de los dos, se evaluará con la nota mínima.

Este laboratorio debe ser entregado el día 1 de Junio del año 2017, hasta las 23:59 hrs. Los descuentos por atraso corren a contar de las 00:59 hrs del día 2 de Junio del año 2017

En caso de dudas o problemas en el desarrollo, comunicarse con su ayudante de laboratorio.