66:20 Organización de Computadoras
Trabajo práctico 2: Memorias caché

1. Objetivos

Familiarizarse con el funcionamiento de la memoria caché implementando una simulación de una caché dada.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 8), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido. Por este motivo, el día de la entrega deben concurrir todos los integrantes del grupo.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo, y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Este trabajo práctico debe ser implementado en C, y correr al menos en Linux.

5. Introducción

La memoria a simular es una caché [1] asociativa por conjuntos de dos vías, de 1KB de capacidad, bloques de 32 bytes, política de reemplazo LRU y política de escritura WT/∼WA. Se asume que el espacio de direcciones es

de 12 bits, y hay entonces una memoria principal a simular con un tamaño de 4KB. Estas memorias pueden ser implementadas como variables globales. Cada bloque de la memoria caché deberá contar con su metadata, incluyendo el bit V y el *tag*.

6. Programa

Se deben implementar las siguientes primitivas:

```
void init()
unsigned char read_byte(int address)
int write_byte(int address, unsigned char value)
unsigned int get_miss_rate()
```

- La función `init()` debe inicializar los bloques de la caché como inválidos y la tasa de misses a 0.
- La función `read_byte(address)` debe retornar el valor correspondiente a la posición `address` si éste se encuentra en el caché. Si no se encuentra, debe cargar el bloque y retornar -1.
- La función `write_byte(int address, unsigned char value)` debe escribir el valor `value` en la posición correcta del bloque que corresponde a `address` y en la dirección `address` de la memoria y devolver 0, si el bloque se encuentra en la caché. Si no se encuentra, debe escribir sólo en la memoria en la posición `address` y devolver -1.
- La función `get_miss_rate()` debe devolver el porcentaje de misses desde que se inicializó el cache, redondeado al entero más próximo.

Con estas primitivas, hacer un programa que lea de un archivo una serie de comandos, que tendrán la siguiente forma:

```
R dddd
W dddd, vvv
MR
```

- Los comandos de la forma "`R dddd`" se ejecutan llamando a la función `read_byte(dddd)` e imprimiendo el resultado.
- Los comandos de la forma "`W dddd, vvv`" se ejecutan llamando a la función `write_byte(int dddd, unsigned char vvv)` e imprimiendo el resultado.
- Los comandos de la forma "`MR`" se ejecutan llamando a la función `get_miss_rate()` e imprimiendo el resultado.

El programa deberá chequear que los valores de los argumentos a los comandos estén dentro del rango de direcciones y valores antes de llamar a las funciones, e imprimir un mensaje de error informativo cuando corresponda.

7. Mediciones

Se deberá incluir la salida que produzca el programa con los siguientes archivos de prueba:

- prueba1.mem
- prueba2.mem
- prueba3.mem
- prueba4.mem
- prueba5.mem

8. Informe

El informe deberá incluir:

- Este enunciado;
- Resultados de las corridas de prueba.
- El código fuente completo del programa modificado, en dos formatos: digital e impreso en papel.

9. Fecha de entrega

La última fecha de entrega y presentación es el jueves 17 de Mayo de 2018.

Referencias

[1] Hennessy, John L. and Patterson, David A., Computer Architecture: A Quantitative Approach, Third Edition, 2002.