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 cuatro vías, de 4KB de capacidad, bloques de 64 bytes, política de reemplazo LRU y política de escritura WB/WA. Se asume que el espacio de direcciones es de

16 bits, y hay entonces una memoria principal a simular con un tamaño de 64KB. Estas memorias pueden ser implementadas como variables globales. Cada bloque de la memoria caché deberá contar con su metadata, incluyendo el bit D, el *tag*, y un campo que permita implementar la política de LRU.

6. Programa

Se deben implementar las siguientes primitivas:

```
void init()
int find_set(int address)
int find_lru(int setnum)
int is_dirty(int way, int setnum)
void read_block(int blocknum)
void write_block(int way, int setnum)
int read_byte(int address)
int write_byte(int address, char value)
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 `find_set(int address)` debe devolver el conjunto de caché al que mapea la dirección `address`.
- La función `find_lru(int setnum)` debe devolver el bloque menos recientemente usado dentro de un conjunto (o alguno de ellos si hay más de uno), utilizando el campo correspondiente de los metadatos de los bloques del conjunto.
- La función `is_dirty(int way, int blocknum)` debe devolver el estado del bit D del bloque correspondiente.
- La función `read_block(int blocknum)` debe leer el bloque `blocknum` de memoria y guardarlo en el lugar que le corresponda en la memoria caché.
- La función `write_block(int way, int setnum)` debe escribir los datos contenidos en el bloque `setnum` de la vía `way`.
- La función `read_byte(address)` debe retornar el valor correspondiente a la posición `address`.
- La función `write_byte(int address, char value)` debe escribir el valor `value` en la posición correcta del bloque que corresponde a `address`.

- La función `get_miss_rate()` debe devolver el porcentaje de misses desde que se inicializó el cache.
- `read_byte()` y `write_byte()` sólo deben interactuar con la memoria a través de las otras primitivas.

Con estas primitivas (más las que hagan falta para manejar LRU y WB/WA), hacer un programa que lea de un archivo una serie de comandos, que tendrán la siguiente forma:

```
R ddddd
W ddddd, vvv
MR
```

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

7.1. Documentación

Es necesario que el informe incluya una descripción detallada de las técnicas y procesos de medición empleados, y de todos los pasos involucrados en el mismo, ya que forman parte de los objetivos principales del trabajo.

8. Informe

El informe deberá incluir:

- Este enunciado;
- Análisis de la performance del programa tal como está presentado, para el caso 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 1 de Noviembre de 2018.

Referencias

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