Práctica 1

Protocolos de coherencia de cachés

1.1. Objetivos

Generales:

- El alumno se familiarizará con la coherencia de cachés para sistemas multiprocesadores.

Particulares:

Al finalizar la práctica el alumno:

- Conocerá dos protocolos de coherencia de caches: MESI y MOESI.
- Podrá medir el impacto de la implementación de los protocolos en el desempeño de cachés.

1.2. Requisitos

- **Conocimientos previos:**
  - Sistemas multiprocesador.
  - Coherencia de cachés.
  - Programación multihilos.
  - Programación en el lenguaje C:
    - Tipos de datos.
    - Estructuras de control.
    - Estructuras y uniones

    - Arreglos y apuntadores.
    - Entrada y salida.

- **Tiempo de realización sugerido:**
  10 horas.
- **Número de colaboradores:**
  Equipos de 2 integrantes.
- **Software a utilizar:**
  - *GCC.*
  - *Pin - A Binary Instrumentation Tool.*

1.3. Planteamiento

En esta práctica se extenderá el simulador elaborado en la práctica anterior, se agreagará la simulación de los accesos a una memoria compartida por dos nucleos, cada uno con un caché. Para asegurar la coherencia, se implementarán dos protcolos: MESI y MOESI, el usuario podrá elegir con cual de ellos se ejecutará la simulación.

1.4. Desarrollo

1.4.1. El simulador

El simulador recibirá un archivo con la traza de memoria de un proceso con dos hilos de ejecución y una configuración para los cachés, ésta será la misma para los cachés de ambos núcleos, el formato será el mismo que el de la práctica anterior (cabe señalar que los cachés deben usar *write back*). Se simulará la ejecución de los accesos a memoria indicados en la traza y se entregarán las mismas estadísticas de la práctica anterior, además una tabla con el número de transiciones de estado del protocolo de coherencia realizadas durante la ejecución del simulador. Es neceario agregar al simulador una abstracción de un *bus* con la finalidad de utilizar *snooping* y se deben agregar los bits necesarios para conservar el estado del protocolo en cada bloque del caché.

1.4.2. Protocolo MESI

El protocolo MESI cuenta con cuatro estados para determinar la validez de un bloque presente en el caché, el estado depende de las otras copias del mismo bloque en otros cachés o si es consistente con la memoria principal. El nombre del protocolo viene del nombre de los estados:

- **Modified**. El bloque sólo se encuentra en el caché actual y se ha modificado, no coincide con la memoria principal.

- **Exclusive**. El caché cuenta con la única copia del bloque y coincide con la memoria principal.
- **Shared**. Existen varias copias del bloque en otros cachés y coinciden con la memoria principal.
- **Invalid**. El bloque fue modificado por otro caché y no es válido.

En las siguientes secciones se describen las acciones que debe llevar a cabo el controlador del caché dependiendo de la acción ocurrida y el estado del bloque.

Read hit

Se lee del caché y no hay cambio de estado. Si el estado del bloque es Invalid se considera como un miss.

Write hit

Modified	Exclusive	Shared
Escribir en caché No cambia el estado	Escribir en caché Cambiar a Modified	Escribir en caché Escribir en bus Cambiar a Modified

Si el estado del bloque es Invalid se considera como un miss.

Write miss

Se anuncia en el bus que se va a modificar el bloque y se escribe el bloque en el caché con estado Modified.

Read miss

- Se anuncia en el bus la lectura.
- Si no hay copias se carga al caché de la memoria principal con estado Exclusive.
- Si algún otro caché tiene el bloque en estado Modified, es necesario esperar a que dicho caché escriba el dato en la memoria principal y se carga al caché de la memoria principal con estado Shared.
- Si algún otro caché tiene el bloque con estado Exclusive o Shared, se carga de la memoria principal con estado Shared.

Remplazo

Si el bloque será reeplazado, el bloque será escrito en la memoria principal sólo si el estado es Modified.

Snooping

Los controladores de caché escuchan en el bus cada uno de los anuncios de lectura y escritura junto con la dirección en memoria solicitada, si se cuenta con una copia del bloque anunciado, debe llevar a cabo las siguientes acciones dependiendo del estado:

	Modified	**Exclusive**	**Shared**
Read	Escribir a mem. pricipal Cambiar estado a Shared	Cambiar a Shared	Sin respuesta
Write	Cambiar estado a Invalid		

1.4.3. Protocolo MOESI

El protocolo MOESI conserva los nombres de estado que el protocolo MESI con la adición de uno nuevo, aunque el significado de los estados no es lo mismo:

- **Modified**. El caché contiene la única copia valida del bloque y ha sido modificada.
- **Owned**. Existen copias de este bloque en otros cachés y no se ha escrito en la memoria principal, éste caché es el dueño (*owner*) del bloque.
- **Exclusive**. Este caché tiene la única copia válida del bloque y coincide con la memoria principal.
- **Shared**. El bloque es una copia exacta de un bloque en otro caché, el dueño (*owner*), el cual puede o no coincidir con la memoria principal.
- **Invalid**. El bloque fue modificado por otro caché y no es válido.

En las siguientes secciones se describen las acciones que debe llevar a cabo el controlador del caché dependiendo de la acción ocurrida y el estado del bloque.

Read hit

Se lee del caché y no hay cambio de estado. Si el estado es Ivalid se considera como un miss.

Write hit

Modified	**Owned**	**Exclusive**	**Shared**
Escribir en caché No cambia el estado	Escribir en caché Escribir en bus Cambiar a Exclusive	Escribir en caché Cambiar a Modified	Escribir en caché Escribir en bus Cambiar a Exclusive

Si el estado es Ivalid se considera como un miss.

Write miss

Se anuncia en el bus que se va a modificar el bloque y se escribe el bloque en el caché con estado Exclusive.

Read miss

- Se anuncia en el bus la lectura.
- Si no hay copias se carga al caché de la memoria principal con estado Exclusive.
- Si algún otro caché tiene el bloque en estado Modified u Owned, es necesario esperar a que éste lo escriba en el bus, posteriormente el controlador de cahé actual lo leerá del bus y lo guardará en su caché con estado Shared.
- Si algún otro caché tiene el bloque con estado Exclusive o Shared, se carga de la memoria principal con estado Shared.

Remplazo

Si el bloque será reeplazado, el bloque será escrito en la memoria principal sólo si el estado es Modified u Owned.

Snooping

Los controladores de caché escuchan en el bus cada uno de los anuncios de lectura y escritura junto con la dirección en memoria solicitada, si se cuenta con una copia del bloque anunciado, debe llevar a cabo las siguientes acciones dependiendo del estado:

	Modified	**Owned**	**Exclusive**	**Shared**
Read	Escribir bloque a bus Cambiar a Owned	Escribir bloque a bus No hay cambio	Sin respuesta Cambiar a Shared	Sin respuesta No hay cambio
Write	Cambiar estado a Invalid			

Como se puede observar, para este protocolo, en el bus no sólo se podrán escribir direcciones y el tipo de acceso a la memoria, si no que también deberá ser posible enviar bloques de cachés a través de éste.

1.5. Entrada

El simulador recibirá por la línea de comandos los siguientes parámetros:

1. Latencia de RAM. Tiempo necesario para traer un dato de la memoria RAM al último nivel de cahé.
2. Una cadena para indicar el protocolo a usar.

3. Archivo con la traza de acceso a memoria.

4. Uno o más archivos con la configuración de los cachés, un archivo por cada caché. El orden definirá los niveles, el primero en aparecer en la línea será el más cercano al procesador y el último en aparecer será el más cercano a la memoria RAM.

Ejemplo:
```
$ simulador 61.5 mesi traza.txt config
```

Formato de archivo de configuración de caché:

Los archivos seguirán la misma configuración que la práctica anterior.

1.6. Salida

El simulador entregará un archivo con los siguientes datos:

- Tasa de aciertos por nivel.
- Tasa de fallos por nivel.
- Tiempo total de ejecución.
- Tiempo promedio de acceso a memoria.
- Una tabla con un conteo de las transiciones de estado hechas en el protocolo usado.

1.7. Variables libres

No hay variables libres para esta práctica.

1.8. Procedimento

La entrega constará de los siguientes elementos:

1. Los archivos de código fuente escritos en el lenguaje de programación C con la solución a los ejercicios. El código debe estar completamente documentado y cumplir con las convenciones.

2. La traza usada para evaluar el desempeño de los cachés.

3. Un reporte con una explicación del diseño del simulador y los análisis de desempeño.

1.9. Ejercicios

1. Escribe un programa multihilo en el lenguaje de programación C. Debes usar dos hilos de ejecución y compartir datos entre ellos. Pon especial cuidado en utilizar las mismas loacalidades de memoria para forzar la ejecución de los protocolos de coherencia.

2. Utiliza pin para obtener la traza de memoria del programa que desarrollaste en el ejercicio 1.

3. Extiende el simulador de la práctica anterior implementando los protocolos de coherencia MESI y MOESI, así como la simulación de ejecución multihilos.

4. Evalúa el desempeño de dos configuraciones de cachés de la práctica anterior con cada protocolo.

5. Elabora un reporte similar al de la práctica anterior.

1.10. Preguntas

1. ¿El desempeño de los cachés se ve afectado por la implementación de los protocolos de coherencia? Compara los resultados obtenidos con los de la práctica anterior.

2. Además de los estados, ¿cuáles son las diferencias entre los protocolos MESI y MOESI? ¿Que impacto tienen en el desempeño?

1.11. Bibliografía

[1] J. Handy. *The Cache Memory Book*. 2nd. Academic Press, 1998.

[2] John L. Hennessy y David A. Patterson. *Computer Architecture: A Quantitative Approach*. 5.ª ed. Morgan Kaufmann Publishers Inc., 2011.

[3] Intel. *Pin - A Dynamic Binary Instrumentation Tool*. Consultado: 16 de abril, 2016. Publicado: 2012. URL: `https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool/`.

[4] Brian W. Kernighan. *The C Programming Language*. 2nd. Prentice Hall Professional Technical Reference, 1988.

[5] David A. Patterson y John L. Hennessy. *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. 5th. Morgan Kaufmann Publishers Inc., 2013.