TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4302 — Arquitectura de Computadores II

Proyecto I

Modelo de protocolo para coherencia de caché en sistemas multiprocesador

Fecha de asignación:	03 de marzo 2023	Fecha de entrega:	10 de abril 2023
Grupos:	1 persona	Profesor:	Luis Barboza Artavia

1. Objetivo

Mediante el desarrollo de este proyecto, el estudiante aplicará los conceptos de coherencia de caché y sistemas multiprocesador en el diseño de un modelo en software del protocolo de coherencia de caché MOESI para un sistema multiprocesador simulado.

2. Atributos relacionados

A continuación se describen los atributos del graduado que se pretenden abordar con el desarrollo del proyecto.

2.1. Diseño (DI)

Capacidad para diseñar soluciones de problemas complejos de ingeniería, con final abierto y diseñar sistemas, componentes o procesos que cumplan con necesidades específicas, considerando la salud pública, seguridad, estándares pertinentes, así como los aspectos culturales, sociales, económicos y ambientales.

El atributo de diseño será evaluado tanto formativamente (reuniones de seguimiento con el profesor) como sumativamente, en especial en la sección de documentación de diseño de los entregables.

3. Descripción general

Los sistemas multiprocesador corresponden a la base funcional de los computadores modernos. Se pueden encontrar en cualquier sistema como en chips, empotrados y *clusters* que permiten una computación de alto desempeño y procesamiento masivo en paralelo. Uno de los mayores retos que genera tener múltiples procesadores es la comunicación entre ellos. Los diseñadores de los computadores actuales han buscado mecanismos para optimizar la capacidad para compartir tareas y recursos entre los diferentes procesadores. La memoria se comparte con los demás procesadores y corresponde a uno de los recursos más importantes, puesto que se compone de los datos

necesarios para ejecutar las instrucciones. Las interfaces de memoria compartida representan un desafío porque múltiples procesadores pueden solicitar un mismo dato en un instante de tiempo. Por lo tanto, el problema de incoherencia de caché todavía se investiga en el área de arquitectura de computadores.

Para este proyecto se deberán aplicar los conceptos de arquitectura de computadores, vista como una combinación de elementos de software y hardware. Se diseñará el modelo de software (plataforma de simulación) de un sistema multiprocesador con **memoria compartida**. Se hará a través de la memoria caché local L1 para cada procesador y una memoria compartida haciendo uso del protocolo de coherencia MOESI.

En el proyecto se desarrollará un acercamiento práctico al diseño de interfaces de memoria compartida y sincronización. Se abarcará el diseño, integración y programación de sistemas computacionales en general.

4. Especificación

Para este proyecto se deberá diseñar una aplicación de software con interfaz gráfica (**no se aceptarán proyectos por medio de terminal**) que modele un sistema multiprocesador como se muestra en la figura 1 con las siguientes características:

1. Tendrá cuatro procesadores.
2. Cada procesador tendrá una memoria caché local L1 con 4 bloques.
3. Estos procesadores se encuentran conectados a una memoria por medio de un bus.
4. La memoria caché L1 es mapeada de forma asociativa por set *two-way*.
5. Cada procesador deberá generar solicitudes de procesamiento o acceso a memoria (lectura o escritura) a diferentes regiones de memoria **de forma aleatoria**.

A continuación se describe a mayor detalle la especificación del proyecto.

4.1. Modelo del sistema multiprocesador

Para el modelo del sistema multiprocesador se hará lo siguiente:

1. Deberá crearse una instancia del procesador de manera independiente para operar de forma paralela (mediante hilos).

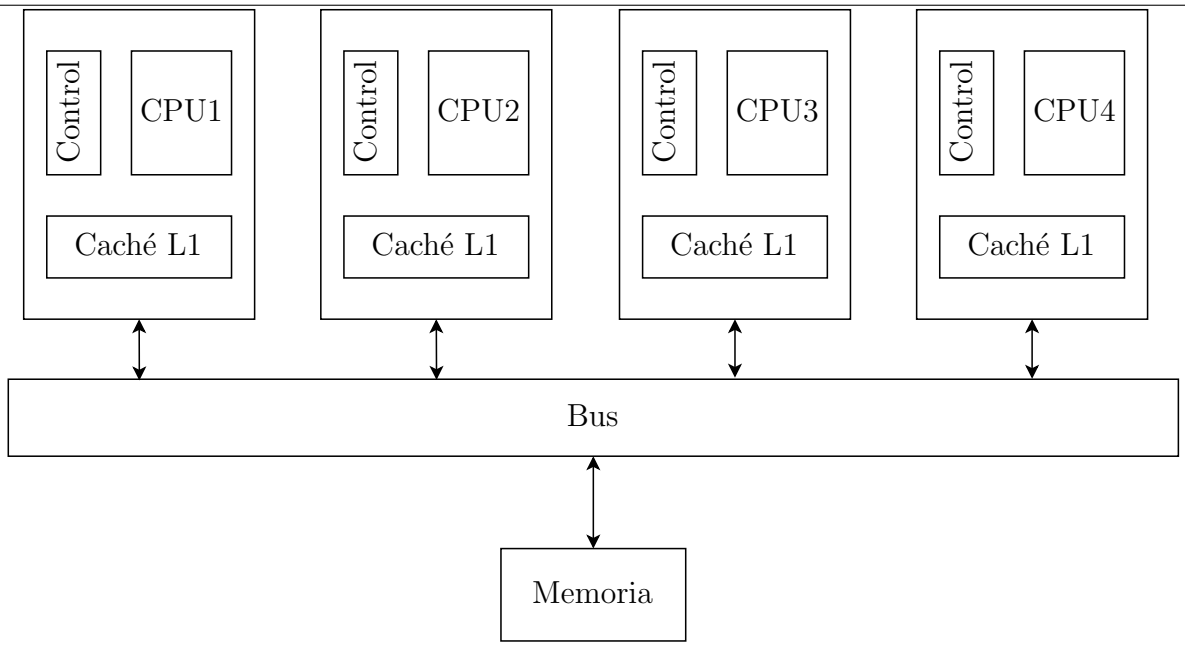

Figura 1: Arquitectura general del sistema.

2. Cada procesador deberá generar individualmente y de manera simultánea junto con los demás procesadores:

   a) Instrucciones de procesamiento (no requiere memoria). Estas instrucciones se llaman *calc*.

   b) Escritura o lectura a uno de los 8 bloques totales de la memoria compartida. Estas instrucciones se llaman *read* y *write* respectivamente.

3. Cada una de las instrucciones debe generarse utilizando una distribución de probabilidad formal construida por el estudiante (binaria, de Poisson, hipergeométrica, etc). **No se permite el uso de bibliotecas. Por ejemplo** `random.poisson(lam=2, size=10)`

4. Una instrucción generada debe tener los siguientes componentes:

   - Número de procesador.
   - Operación: *read*, *write* y *calc*.
   - En el caso de *read* se debe indicar la dirección de memoria (binario).
   - En el caso de *write* se debe indicar la dirección de memoria (binario) y el dato a escribir (hexadecimal), respectivamente.

5. Una vez obtenida la instrucción, la dirección del bloque de memoria también será asignada de manera aleatoria con una distribución previamente documentada y justificada.

Una instrucción tiene el siguiente formato de ejemplo:

```
P0: READ 0100
P1: CALC
P3: WRITE 1010;4A3B
P4: CALC
```

El modelo del sistema deberá proveer una base de temporización para la generación de los eventos de procesamiento, lecturas y escrituras a definir por cada estudiante. Dicha base debe permitir la correcta visualización de todas las acciones generadas por el sistemas multiprocesador y sistema de coherencia de caché.

En todo momento deberá mostrarse la acción de cada núcleo, así como el bloque de memoria que está utilizando. Cada procesador será identificado por su número único.

4.2. Modelo de memoria principal

La memoria principal será unificada y compartida, comunicándose con los procesadores mediante un único bus. Las funciones del modelo memoria principal serán:

1. Actualizar el contenido compartido de los bloques en escrituras (según política de *write-back*).

2. Permitir la visualización de los 8 bloques en todo momento.

El sistema deberá simular las condiciones propias de pared de memoria. Tendrá un retardo propio de lectura o escritura a memoria, cuando corresponda. El tiempo debe ser proporcional a la frecuencia elegida anteriormente, de manera que exista el problema de que el procesador es más rápido que la memoria.

Al iniciar la aplicación, el contenido de los bloques de memoria deben ser 0. Cada bloque es de 16 bits y debe ser representado en hexadecimal.

4.3. Modelo de memoria caché y sistema de coherencia

Cada procesador tiene su caché L1, se pueden almacenar 4 bloques en cada una y posee correspondencia asociativa por set *two-way*.

Esta memoria servirá para almacenar cuatro bloques que requiera el procesador. La información que debe contener es:

- Número de bloque.
- Estado de coherencia: M (modificado), S (compartido), I (inválido), E (exclusivo) y O (dueño).
- Dirección de memoria.
- Dato de 16 bits en hexadecimal.

Se asumirá que si el dato es referenciado por primera vez, este no se encuentra en caché. La caché deberá generar alertas de *misses* tanto por escritura como por lectura. Además deberá permitir la visualización del contenido de los 4 bloques, por cada procesador, en todo momento.

Adicional al modelo de la caché, deberá diseñarse un modelo del sistema de coherencia, dentro del controlador, que deberá asegurar la coherencia entre todas las cachés. El modelo deberá incorporar una política de invalidación para las cachés de los demás procesadores. Ante una actualización en alguna caché, o un *miss* por invalidación, el sistema de coherencia deberá controlar la lectura del bloque correspondiente desde memoria principal, y la escritura hacia las demás cachés.

Para este diseño, se debe basar en el protocolo de monitoreo MOESI. Debe quedar claro el tipo de protocolo utilizado y su descripción detallada. El controlador, además, deberá asegurar el funcionamiento correcto de la correspondencia de la caché.

Al iniciar la aplicación la caché estará fría, es decir, el contenido de los bloques de caché debe ser 0. Deberá realizar estos desaciertos obligatorios.

4.3.1. Interfaz gráfica

El sistema debe visualizarse por medio de una interfaz gráfica con el fin de observar el comportamiento de cada uno de los procesadores al generar las instrucciones. La temporización elegida debe permitir la correcta visualización de todas las acciones generadas por el sistema multiprocesador y sistema de coherencia de caché.

Los elementos que deben visualizarse son:

- Identificador de cada procesador.
- Información de la caché L1 detallada en la sección anterior para cada procesador.
- Última instrucción generada, así como la instrucción generada para cada procesador.

- Contenido de la memoria.
- Alertas de *miss* tanto de lectura como de escritura.

El sistema a nivel temporal tendrá dos modos:

1. Paso a paso (solo ejecuta el siguiente ciclo).
2. Ejecución continua (no se detiene hasta que el usuario lo detenga con pausa).

Se usan pausas para poder distinguir de mejor manera la transición de los estados de los diferentes bloques de caché. Además, el usuario puede agregar una instrucción a ejecutarse en el siguiente ciclo. Para ello, cuando el sistema esté en pausa, se podrá elegir un procesador e indicar el tipo de instrucción, así como la lectura o escritura con sus respectivas direcciones y dato en caso de que aplique. Sólo se podrá agregar una instrucción en modo pausa y se verá reflejada luego de ejecutar la instrucción actual.

5. Metodología de trabajo

El proyecto debe seguir los siguientes aspectos de desarrollo, sino, la parte funcional no será calificada y obtendrá nota de cero:

1. Utilice una cuenta de repositorio gratuita.
2. Cree un repositorio con el siguiente nombre: `<user_id>_computer_architecture_2_2023`. El `user_id` estará compuesto por la primera letra del nombre y el apellido. Por ejemplo, para el estudiante Luis Barboza, el nombre del repositorio será:

   `lbarboza_computer_architecture_2_2023`.
3. Si el repositorio es privado, proporcione acceso a `luisbarbozaCE` (GitHub y GitLab).
4. El repositorio de Git contendrá dos ramas principales: `master` y `development`.
5. Inicialmente, la rama de `development` se crea a partir del `master`.
6. Al trabajar en un proyecto, el estudiante debe crear una nueva rama de trabajo desde `develop` y cuando la función esté lista, la rama debe fusionarse para `develop`. Cualquier corrección o modificación adicional después de `merge` debería requerir que se repita el proceso (es decir, crear la rama desde `develop` y fusionar los cambios más tarde). Una vez que el código de desarrollo esté listo, se fusionará con `master` y se debe crear una `tag`. El proceso se describe en la siguiente Figura 2.

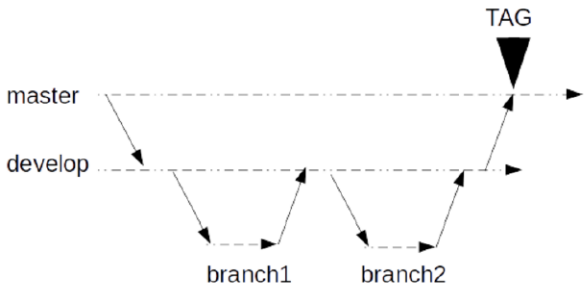

Figura 2: *Git workflow*

Adicionalmente se coloca este enlace recomendado.

7. Después de haber realizado algunos proyectos la rama `master` debe verse así:

   - `master/`
     - `proyecto_1`
     - `proyecto_2`
     - ...
   - ...

   Donde cada directorio de `proyecto_x` contiene todos los entregables para cada proyecto.

   No es permitido realizar todo el trabajo en un solo commit, es decir, que realice el trabajo de forma local y solo suba el último entregable en el repositorio. Si no, obtendrá nota de cero. Debe mostrar avance incremental (se revisarán estadísticas).

5.1. Notas adicionales

- El desarrollo de este proyecto es individual.
- El lenguaje de programación o herramientas de desarrollo podrá ser elegido abiertamente por cada estudiante.
- Todo diseño deberá tener al menos 2 propuestas detalladas adecuadamente y comparadas según criterios.

6. Entregables

Como entregables en este proyecto se evaluará lo siguiente:

- Presentación funcional completa (65 %): cada estudiante deberá demostrar en una sesión (previa cita con el profesor) las diferentes pruebas sobre el sistema. El profesor evaluará las pruebas según rúbrica correspondiente. En la sesión se harán preguntas relacionadas sobre cualquier etapa del sistema. Antes de la revisión deberá colocar el enlace del Git.

- Presentación del *paper* y resultados (15 %): Cada persona deberá grabar y entregar un video de **4:30 minutos a 5:30 minutos**, presentando la idea de su solución (puede utilizar diapositivas o algún otro medio de referencia). Debe considerar que el público meta de su presentación no tiene necesariamente el *background* técnico, por lo que deberá exponer de forma clara lo que se ha realizado, así como los resultados más importantes de su diseño y conclusiones. Puede utilizar el canal de Youtube "Two Minute Papers", como referencia. Por motivos de acreditación del programa, se recomienda vehemente que el enlace proporcionado esté disponible por cualquier persona que tenga acceso a él, hasta un año después de entregada esta evaluación.

- Documentación de diseño (20 %): La documentación del diseño deberá contener las siguientes secciones:

    1. Listado de requerimientos del sistema: Cada estudiante deberá identificar las necesidades y los requerimientos de un problema complejo de ingeniería considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.
    2. Elaboración de opciones de solución al problema: Para el problema planteado deberán documentarse al menos dos opciones de solución. Cada solución deberá ser acompañada de algún tipo de diagrama. **Estas opciones de solución no deben ser fácilmente descartables y deben llevar un análisis objetivo con base en criterios técnicos o teóricos.**
    3. Valoración de opciones de solución: Se deberán valorar alternativas de solución para un problema complejo de ingeniería que cumplan con necesidades específicas, considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.
    4. Selección de la propuesta final: Se deberá seleccionar una propuesta final de las opciones de solución, de acuerdo con los criterios de comparación.

5. Diseño de la alternativa seleccionada: Se deberá documentar completamente el diseño final seleccionado considerando la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario. Para el caso de este proyecto esto incluye: descripción del protocolo diseñado, diagrama de bloques del modelo sistema multiprocesador, diagrama de bloques del computador (procesadores + memoria y aplicación), diagramas propios de diseño de sofware aplicables (de flujo, clases, composición, UML, patrones de diseño, etc) y descripción de algoritmo propuesto y distribuciones de probabilidad implementadas.
6. Validación del diseño: se deberá validar el diseño final de acuerdo con los requerimientos, la salud y la seguridad pública, el costo total de la vida, el carbono neto cero, así como aspectos relacionados con recursos, culturales, sociales y ambientales según sea necesario.

Si tienen dudas puede escribir al profesor al correo electrónico. **Los documentos serán sometidos a control de plagios**. La entrega se debe realizar por medio del TEC-Digital en la pestaña de evaluación. No se aceptan entregas extemporáneas después de la fecha de entrega a las 11:59 pm como máximo. Las revisiones se realizarán el 11 de abril.