Instituto Tecnológico de Costa Rica

Escuela de Administración de Empresas

Curso: CE-4302:
Arquitectura de Computadores II

Proyecto #1 – Documento de diseño.

Realizado por:

Gerardo Zeledón Martínez, 2016243863

Profesor:

Ing. Luis Barboza Artavia

Fecha: martes 2 de mayo, 2020

Requerimientos del sistema:

Para un correcto funcionamiento del sistema, de acuerdo con la especificación del proyecto, se debe cumplir que:

- El sistema debe poseer dos procesadores de los cuales cada uno tiene 2 núcleos.
- Cada núcleo del procesador debe generar instrucciones de manera asincrónica.
- Cada instrucción debe ser atendida de manera simultánea entre todos núcleos.
- El sistema debe usar el protocolo MSI para manejar la coherencia entre caché.
- Cada núcleo debe tener una cache L1 propia, la cual deba tener dos bloques de memoria con correspondencia directa.
- Cada procesador debe tener su propia cache L2, la cual tendrá cuatro bloques de tamaño con correspondencia directa y además será compartida entre los dos procesadores.
- Cada procesador deberá tener un controlador el cual se encargará de la comunicación entre la L2 propia y la del otro procesador.
- Las instrucciones generadas deben ser de al menos tres tipos “WRITE”, “CALC”, “READ”.
- Las instrucciones deben ser generadas de manera aleatoria usando una distribución formal.
- El sistema debe tener una memoria principal única de 16 bloques de tamaño, la cual tendrá un único bus encargado de transportar las señales provenientes de los procesadores.
- El sistema deberá brindar una interfaz en la cual se podrá mostrar la información de tal manera que sea fácil de entender para los usuarios.

Propuestas para la solución del problema:

1. Propuesta Write back usando Python:

Para esta propuesta se propone usar Python como lenguaje para la implementación de la solución aplicando además la política de escritura Write back el cual, como se observa en la figura de la imagen 1 el cual consiste en hacer un Write en memoria hasta que se dé una cache miss.

Se escoge Python debido a que es un lenguaje el cual permite la implementación de programación orientada a objetos además de que posee una serie de herramientas necesarias para el desarrollo de este problema, tales como el manejo de los hilos, creación de la interfaz gráfica usando Tkinter, fácil creación de código gracias a la inferencia de tipos. Además de que, al ser un lenguaje muy utilizado en la actualidad, existe mucha información y documentación la cual puede ser consultado en caso de ser necesario.

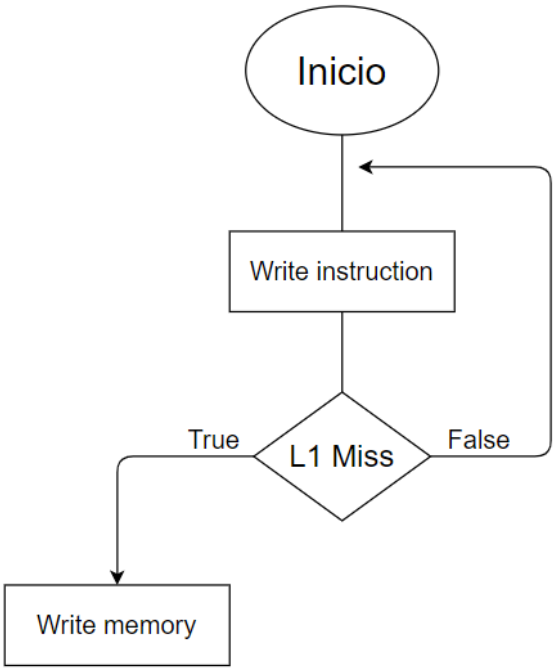

Imagen 1. Diagrama de flujo del Write back.

2. Propuesta Write through usando Java:

Para la segunda propuesta de implementación del sistema se presenta usar la política de escritura Write through la cual, a diferencia de la mostrada en la propuesta anterior, esta política consiste en que justo cuando se actualiza un valor de la cache, este se actualiza también en la memoria principal tal y como se observa en el diagrama de la imagen 2.

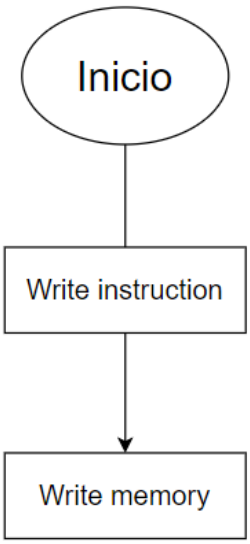

Imagen 2. Diagrama de flujo del Write through.

Además, se determina usar el lenguaje de programación Java por razones similares a Python, es un lenguaje fácil de usar con una amplia documentación detrás. Posee herramientas para la creación de interfaz gráfica como pueden ser swing o JavaFX las cuales son sencillas de usar y se adaptan de buena manera a los requerimientos del problema.

También es importante mencionar que estas propuestas de tal manera que se puede escoger hacer Write back con java y Write through con Python ya que ambos lenguajes poseen herramientas muy similares.

Comparación de opciones de solución:

1. Desde el punto de vista de la implementación:

Tomando en cuenta la dificultad de implementación de los algoritmos para las dos políticas de escritura mencionadas en las dos propuestas; el desarrollo del Write back es más complicado que el del Write through ya que el hecho de que se tenga que esperar al que exista una miss en cache implica que se tienen que hacer más comparaciones y tener encuenta más casos a la hora de la ejecución.

Por otro lado, con el método del Write through al solo tener que escribir en memoria al momento de que se escribe en cache simplifica la implementación ya que se sigue un flujo directo y no se deben hacer tantas comparaciones además de que se reduce la incoherencia entre las cache.

2. Desarrollo de la interfaz gráfica:

Con respecto al desarrollo de la interfaz gráfica, Python brinda la herramienta Tkinter, la cual es compatible tanto para sistemas Linux como Windows sin embargo hay determinadas funciones que cambian ente un sistema operativo u otro. Esto es un factor importante para tomar en cuenta.

En el caso de Java se tienen las bibliotecas swing o JavaFX las cuales tienen la particularidad de que se pueden desarrollar las vistas usando el estándar XML lo cual facilita la creación de las vistas, además por si no fuera poco, existen herramientas con los cuales se puede crear interfases de usuario simplemente con “drag & drop” haciendo que la creación de las interfases sea más rápida.

3. Complejidad del lenguaje de programación:

Los dos lenguajes de programación propuestos son realmente sencillos y el sistema planteado perfectamente se podría realizar en ambos, ya depende de los gustos del programador decantarse por uno u otro.

Ambos cuentan con herramientas para el manejo de hilos, mutex, programación orientada a objetos, manejo de archivos, entre otros.

Selección de la propuesta final:

Tomando en cuenta todas las propuestas descritas en los puntos anteriores se decide utilizar la propuesta de hacer el sistema utilizando la política de escritura Write through en el lenguaje java. La decisión se basa en los siguientes puntos:

- La facilidad de crear la interfaz con Scene Builder de JavaFX.
- La estructura completa de java para la programación orientada a objetos.
- La amplia documentación del lenguaje.
- La facilidad en la implementación del algoritmo del Write through.
- La fácil declaración y manejo de hilos en Java.
- El fácil manejo de archivos para crear los logs.

Implementación del diseño:

Para desarrollar el siguiente sistema será necesario programar las clases del diagrama de la imagen 3. Como se puede observar el sistema contara con una clase dedicada a cada parte por separado. A continuación, se detallará una a una todas las clases a profundidad para explicar su aporte al sistema y el porque de todos los atributos y métodos que la componen.

1. **Core:** esta clase es la abstracción de los núcleos de los procesadores, en ella se generarán las nuevas instrucciones y es la que tendrá la memoria L1 como una matriz de Strings. Es el nivel mas bajo del sistema y la base para el funcionamiento en general. En caso de que se dé una escritura o una lectura, esta clase levantara unas banderas, las cuales avisaran a un nivel superior que el núcleo necesita atención por parte del procesador.

2. **CPU:** esta clase es la encargada de guardar la matriz L2, instanciara dos objetos de tipo Core. Al igual que la clase Core, esta clase posee banderas que servirán para avisar que se necesita atención por parte del sistema para atender al procesador. Existe una bandera en especifico la cual le hace saber

al procesador si un dato leído proviene de la cache L2 del otro procesador o bien proviene de la memoria principal. Finalmente, esta clase posee una serie de métodos los cuales serán los encargados de atender las banderas provenientes de los núcleos.

3. **Sistema:** Como su nombre lo indica, esta es la clase principal del sistema la cual se encargará de manejar a los dos procesadores que tendrá instanciados, almacenará los datos de la memoria principal y proveerá métodos para atender las banderas que generan los procesadores como solicitudes de lectura o escritura hacia la memoria principal.

4. **Instruction Generator:** Aquí es donde se generan las instrucciones de manera completamente aleatoria usando una distribución binomial.

5. **Main:** Esta es la ventana principal de la aplicación, aquí se define el tamaño de la ventana, icono y se referencia al archivo con extensión fxml el cual posee todas las vistas de la interfaz de usuario. Al ser esta la única ventana de la aplicación, en ella se instancia la clase sistema, la cual proveerá la información necesaria para poder ser mostrada en pantalla.

6. **Controller:** La clase controller es la encargada de manejar todos los objetos que se muestran en pantalla, además de poseer funciones para alterarlos. Dichos métodos serán utilizados para actualizar las tablas y textos que se muestran en la ventana.

Es importante mencionar que las clases Core, CPU y sistema heredan de la clase Thread de java lo cual permite que al momento de ser instanciadas y ponerse a ejecutar estas lo hacen en un nuevo hilo, permitiendo la ejecución asincrónica de los mismos.

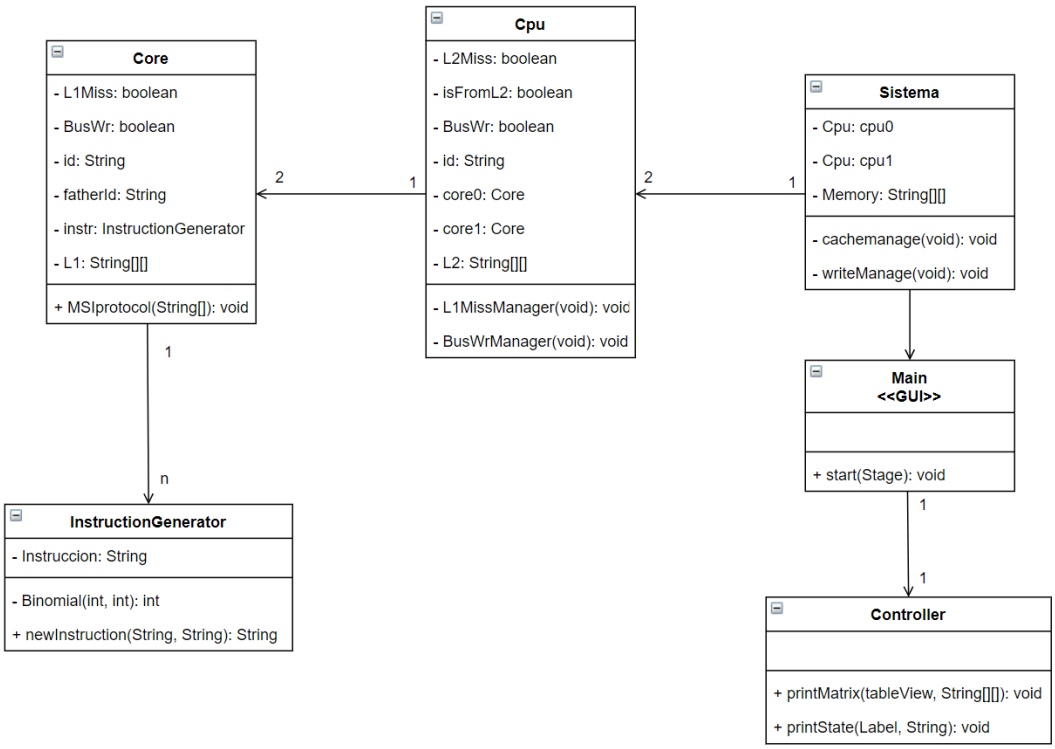

Imagen 3. Diagrama de clases del sistema.

En cuanto a la estructura general del sistema, en los diagramas de las figuras 4 y 5 se puede observar de manera gráfica como está la jerarquía del sistema. Se puede observar cómo están distribuidos los diferentes objetos del diagrama de clases.

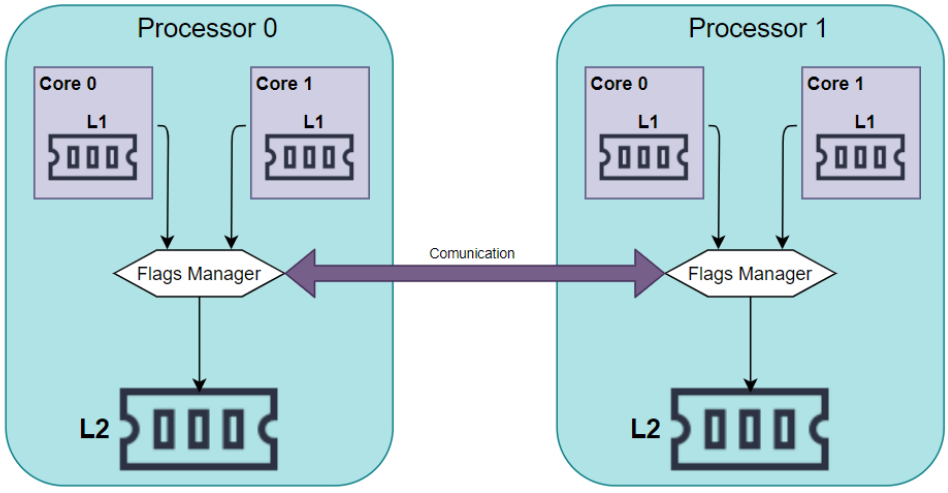

Imagen 4. Diagrama de bloques del multiprocesador.

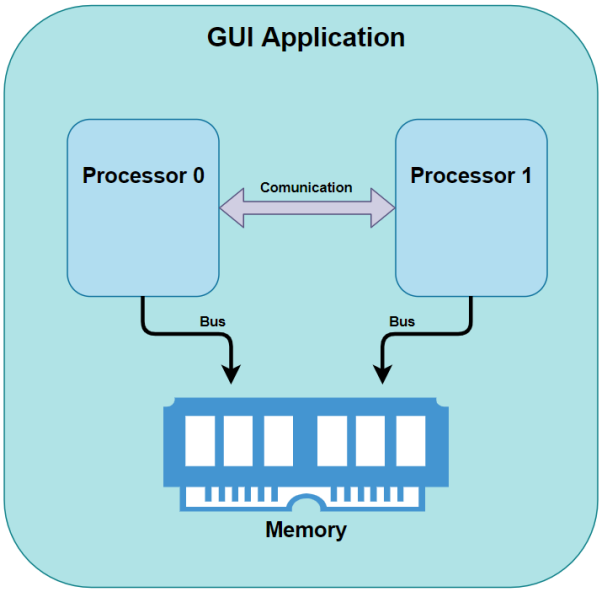

Imagen 5. Diagrama de bloques del Sistema.

El tratamiento que se le va a dar a las instrucciones será tomando en cuenta el protocolo de coherencia de cache MSI y la política de escritura Write through. Tomando en cuenta estas primicias se desarrolló el diagrama de flujo de la imagen 6 en el cual primero se evalúa si la instrucción corresponde a un Write, luego se evalúa si la operación es un Read y de ultimo un Calculo ya que este no representa ningún inconveniente relacionado con la memoria o las cache.

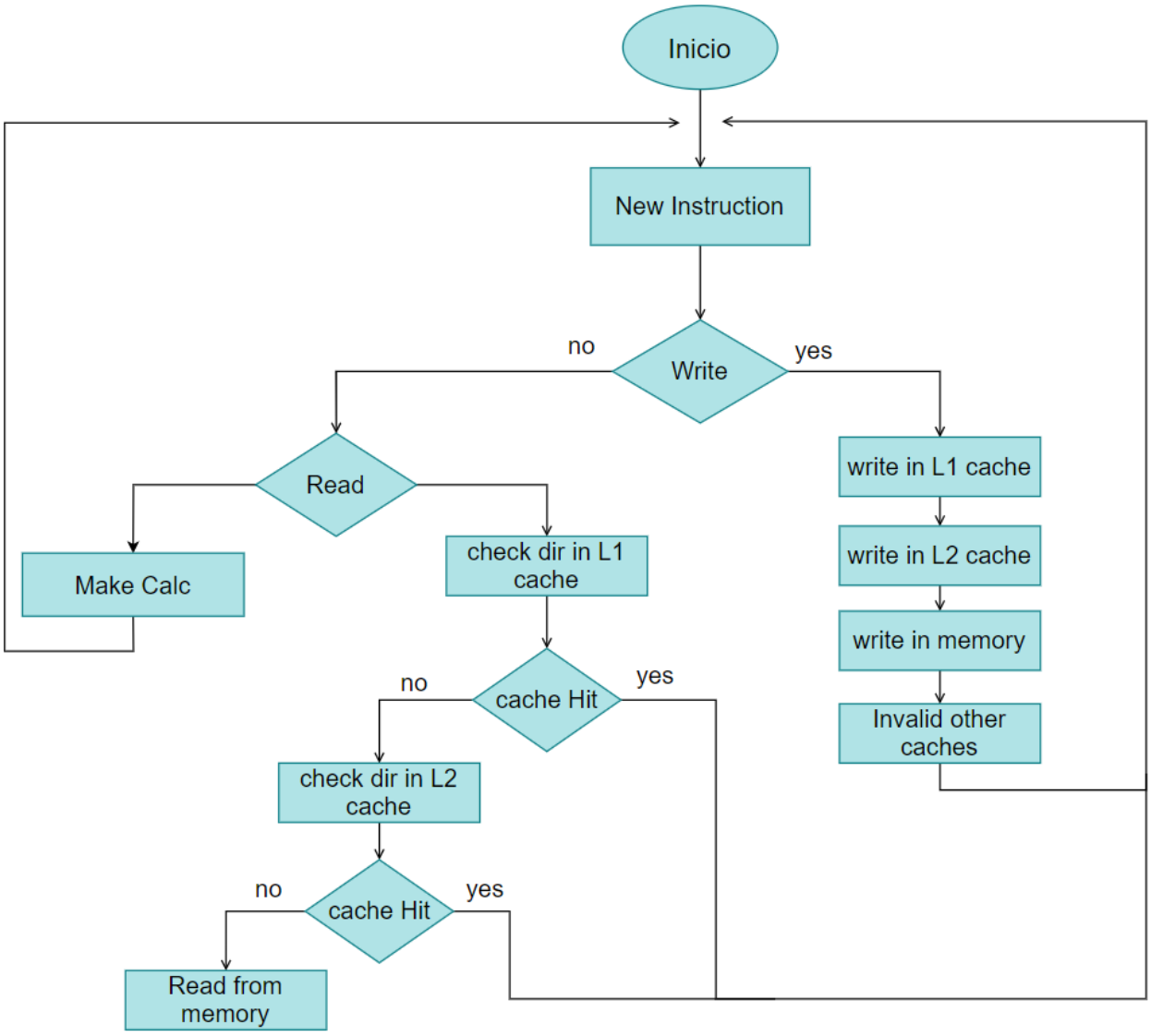

Imagen 6. Diagrama de clases del Sistema.