Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

CE-4302: Arquitectura de Computadores II

(Course: CE-4302 Computer Architecture II)

Proyecto #1 - Protocolos de coherencia en sistemas multiprocesador

(Project #1)

Realizado por:

(Made by:)

Juan D. Esquivel Rojas, 2016167796

Profesor:

(Professor:)

Ing. Luis Barboza Artavia

Fecha de entrega: Cartago, 2 Junio, 2020

(Due Date: June 2nd, 2020)

Requerimientos del Sistema

Según la especificación del proyecto brindada por el profesor, y teniendo en cuenta diversas dudas y preguntas generadas a partir del mismo, se enlistan los siguientes requerimientos del sistema:

- El sistema a desarrollar debe presentar un módulo que genere instrucciones de lectura, escritura y cálculos, con su tipo, direcciones y datos obtenidas de manera aleatoria.
- La generación aleatoria de las instrucciones debe de realizarse utilizando una distribución de probabilidad formal (Binaria, de Poisson, hipergeométrica, binomial, etc).
- Cada instrucción se debe generar de manera paralela e independiente.
- El sistema debe contar con dos chips en los cuales se utilizan dos procesadores cada uno.
- El sistema a diseñar debe poseer dos niveles de caché: L1 y L2.
- La memoria caché L1 a implementar debe tener dos bloques.
- Cada procesador debe tener a su disposición una memoria caché L1.
- La memoria caché L2 debe ser implementada con cuatro bloques.
- La memoria caché L2 debe estar implementada para cada chip dentro del sistema.
- Todas las memorias caché implementadas debe de ser implementadas con correspondencia directa.
- Para todo el sistema a diseñar, se debe crear una única memoria principal.
- La memoria caché L2 debe ser compartida por cada uno de los chips a implementar.
- El funcionamiento de la caché L1 debe ser diseñado utilizando el protocolo de monitoreo para controlar la coherencia.
- La memoria caché L2 se implementará utilizando el protocolo de directorios para controlar la coherencia de la misma.

- El sistema a crear debe permitir la visualización de los eventos que sucedan dentro del mismo y todas las tablas de memorias utilizadas.
- El sistema por diseñar debe de registrar todas los eventos realizados por el mismo en un archivo .log.

Propuestas de diseño

Una vez analizados todos los requerimientos del sistema planteados anteriormente, se crean dos propuestas de diseño para la solución al problema, tomando en cuenta aspectos como: la política de escritura que se implementará, el protocolo de coherencia, el lenguaje de programación a utilizar, su facilidad de elaboración y la estructura a diseñar del sistema como tal. A continuación, se presentan las propuestas planteadas:

Propuesta 1

Según los aspectos planteados para escogencia de solución al problema, se diseña la propuesta 1.

Para la propuesta 1 se implementa como política de escritura el sistema *“write-back”.* Al utilizar esta política, cuando el procesador ejecuta una instrucción de tipo escritura, se escribe únicamente en la memoria caché del mismo. Esto provoca que si el otro procesador dentro del mismo chip realiza otra instrucción pero del tipo de lectura sobre la misma dirección que el procesador anterior escribió, este último deberá de escribir en la memoria caché L2. Ocurre lo mismo se fuera el caso contrario e inclusive en la memoria caché L2, ya que si la memoria L2 vista desde el otro chip necesita un dato que la otra L2 modificó, esta deberá de escribir el dato actualizado en la memoria principal. Para un mejor entendimiento de la implementación de esta política dentro de la solución planteada, se adjunta la figura 1.

Para esta propuesta de diseño se utilizará como lenguaje de programación JavaScript. Este lenguaje actualmente se encuentra en un auge con respecto a cuantos programadores lo utilizan, ya que presenta una serie de facilidades en cuanto a la implementación y su uso dentro de muchos tipos de APIs y bibliotecas como lo son React, AngularJS, entre otros. Otra de las facilidades que se encuentran al utilizar JavaScript, es la facilidad en la creación de interfaces gráficas de programas, utilizando poco tiempo.

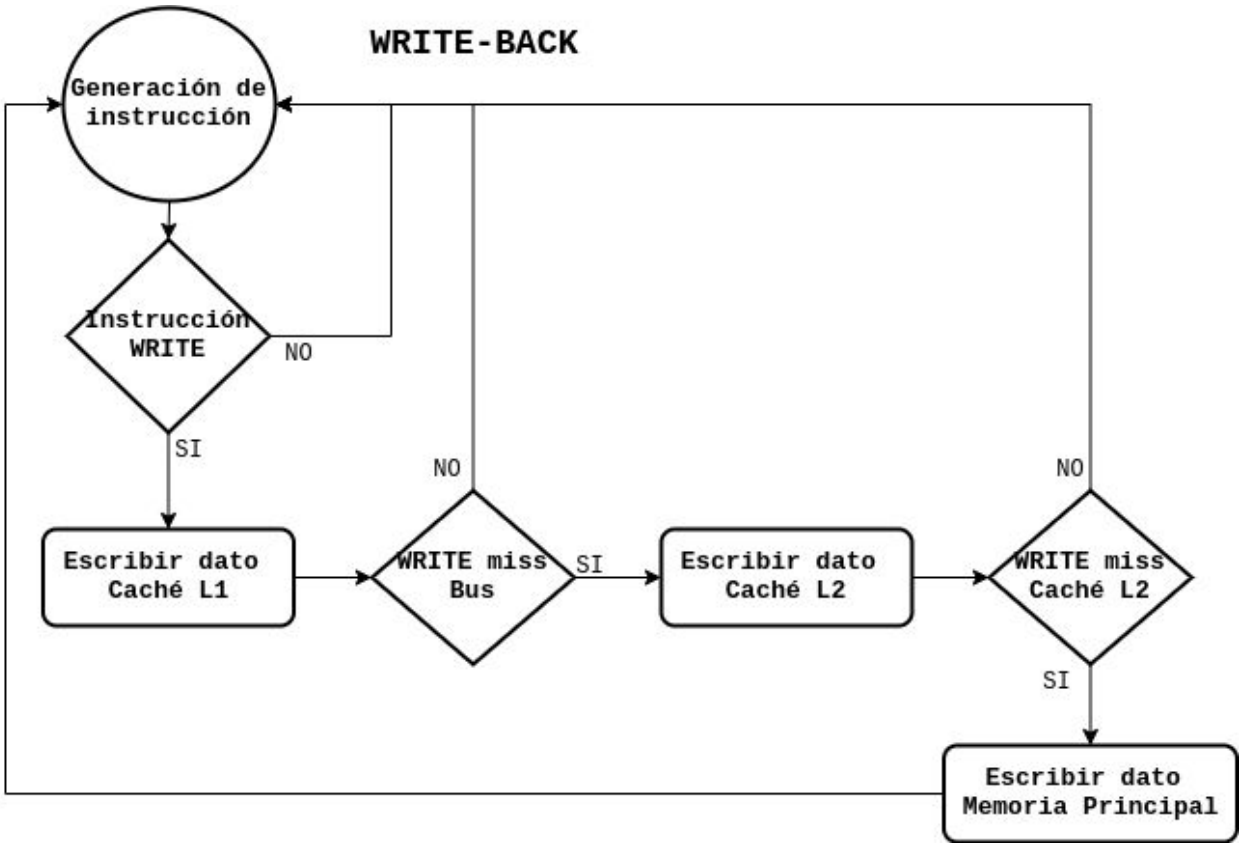

Figura 1. Diagrama de política de escritura Write-Back.

Propuesta 2

Como parte de la segunda propuesta, se diseña el sistema a base de la otra política de escritura, llamada "*Write-Through"*. El funcionamiento de esta política es más sencillo que el *Write-Back*: cuando el procesador genera una instrucción WRITE

esta se escribe en las memorias caché y en memoria principal por igual, generando que los datos en las memorias caché no se encuentren bajo problemas de coherencia y siempre estén actualizados con respecto a la memoria principal. Como complemento a la explicación del funcionamiento de *Write-Through,* se adjunta la figura 2.

Para esta solución se propone como lenguaje de programación Java. Uno de los lenguajes de programación mayor utilizados desde su creación. Java es un lenguaje que permite la programación utilizando el paradigma de programación orientado a objetos, lo que permite una mejor abstracción de los módulos a crear para la solución al proyecto mediante la utilización de objetos. Con base a esto, Java también posee la facilidad de implementación de hilos y herramientas de creación de interfaz gráfica de manera rápida y sencilla como lo es Java AWT y Swing.

Parte de la escogencia de Java para esta propuesta es el conocimiento y control que se posee en el mismo, para la solución de proyectos. Dicho conocimiento y control al usar Java cae en la experiencia al usarlo en anteriores cursos de Ingeniería en Computadores.

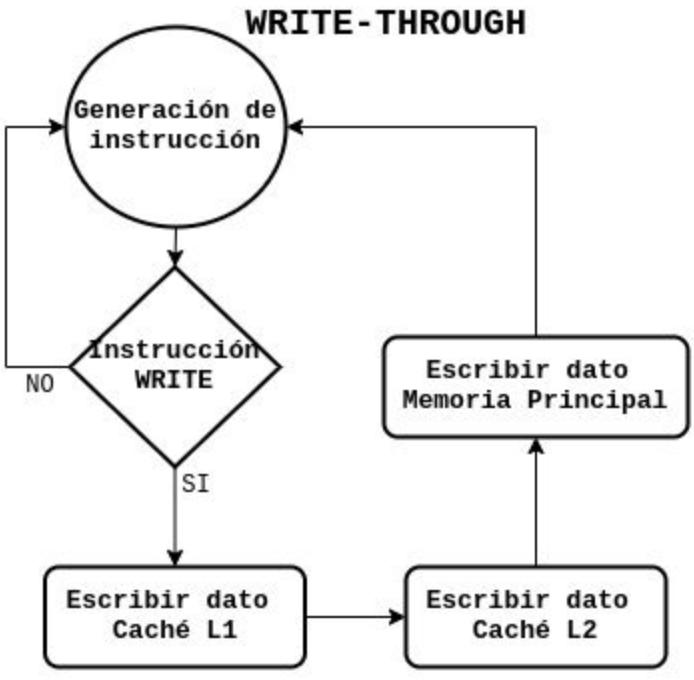

Figura 2. Diagrama de política de escritura Write-Through.

Comparación de propuestas de solución

Para la comparación entre las propuestas de solución al problema expuesto, se desarrollan una serie de criterios de escogencia tomando en cuenta los requerimientos del sistema ya presentados y otros aspectos que sean aplicables a diferentes áreas dentro de lo que es Ingeniería en Computadores, ya que ambas propuestas están diseñadas para que puedan abarcar y aplicar todos los requerimientos del sistema dentro del proyecto.

Con respecto a la facilidad de diseño e implementación del sistema multiprocesador, Java tiene un gran ventaja ya que en comparación con JavaScript, el sistema a desarrollar necesita de un manejo de múltiples hilos en funcionamiento simultáneo, lo cual en Java puede ser un trabajo de menos costo, gracias a sus funciones dentro de la biblioteca Thread, el cual nos permite un manejo más controlado de los mismos. De la misma forma, el utilizar Java para el desarrollo de la solución, puede ser beneficioso por su paradigma de programación orientada a objetos, ya que se puede hacer una abstracción más profunda a un alto nivel, con base a los módulos que se necesitan para la solución del sistema. Además el programar bajo este paradigma podría darnos mejor una idea de lo estamos creando en determinado tiempo.

Por otra parte, el usar JavaScript proporciona ventajas como una mayor flexibilidad y libertad a la hora de programar, ya que no es tan cerrado y poco permisivo como lo es Java, provocando así que se de una limitación en el aspecto de implementación de diversas técnicas de programación.

Otro aspecto importante a tomar en cuenta dentro de la escogencia de la propuesta, es la facilidad de implementación de la interfaz gráfica del sistema. En este caso JavaScript tiene una clara ventaja ya que proporciona una serie de plugins y bibliotecas como React que permiten crear interfaces gráficas de manera más rápida y sencilla, además de que visualmente se tendrá un mejor resultado.

Con respecto a las políticas de escrituras propuestas anteriormente, es claro que la implementación de Write-Through es mucho más sencilla que utilizar el Write-Back, ya que no se requiere implementar un sistema para la sincronización de procesadores. Sin embargo, si se tuviera que implementar una solución de un sistema real (utilizando hardware), el implementar Write-Back sería la mejor opción ya que utiliza menos el bus de comunicación, aspecto que es de vital importancia en la eficiencia de un multiprocesador real.

Según la especificación y los requerimientos expuestos, para el desarrollo de este proyecto no se requiere tomar en cuenta aspectos aplicables a temas ambientales, económicos, culturales y sociales.

Selección de propuesta final

Tomando en cuenta los aspectos mencionados y comparados anteriormente, y planteando la prioridad de los mismos según su importancia dentro de proyecto se selecciona la propuesta final. La propuesta 2 fue seleccionada como el diseño final de la solución al proyecto.

Se escogió esta propuesta ya que el usar la programación orientada a objetos para la creación del sistema del multiprocesador y las funciones de todos los componentes necesarios, puede ser lo más seguro y ventajoso a la hora de definir una estructura del proyecto mismo. También se cuenta con una mayor experiencia trabajando con este lenguaje y además se encuentra una mayor documentación de Java que de cualquier otro lenguaje .

De la misma forma, el escoger utilizar la política de escritura Write-Through es mucho más beneficioso debido a su fácil implementación, en comparación con el Write-Back, ya que para el caso de este proyecto, no se requiere implementar hardware dentro de la solución, por lo que no se toma en cuenta.

Implementación del Diseño

Protocolo de coherencia

Para la implementación de la coherencia de caché del sistema a desarrollar, se dará uso al protocolo MSI, el cual cuenta con menos estados que otros protocolos como el MESI y MOESI, y por lo tanto es ventajoso a la hora de desarrollar y programar el sistema, ya que al ser una simulación de un sistema multiprocesador no es necesario tomar en cuenta más estados.

De todas formas, el utilizar el protocolo de coherencia MSI, cumple con todos los requerimientos del sistema expuestos anteriormente. A continuación, se presenta el diagrama de estados del protocolo MSI utilizado.

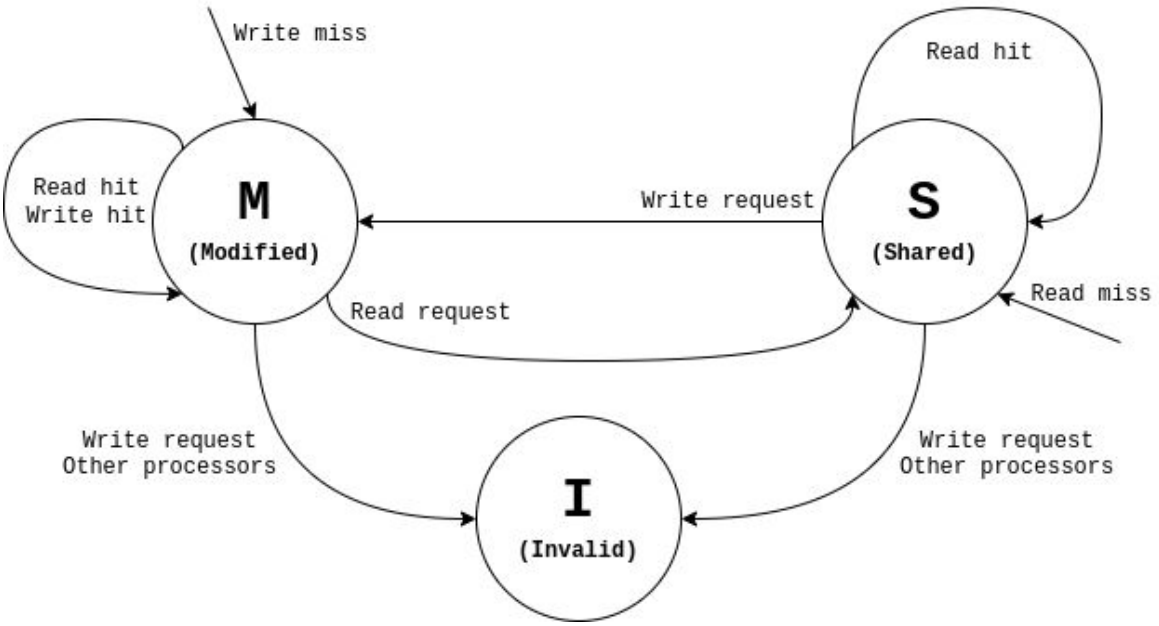

Figura 3. Diagrama de estados de protocolo de coherencia utilizado.

Diagrama de bloques de sistema multiprocesador

Como parte del sistema multiprocesador a desarrollar, se crea un diagrama de bloques de alto nivel, en el cual se muestran las diversas unidades a desarrollar y su conexión entre las mismas para su correcto funcionamiento.

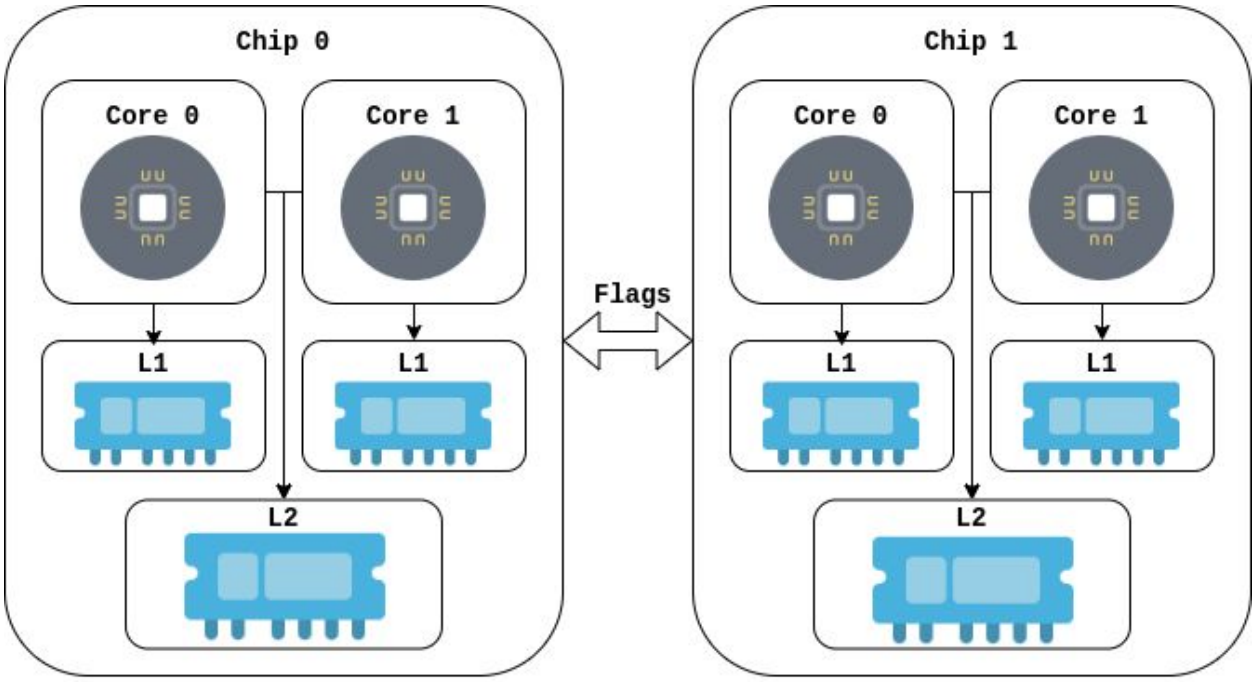

Figura 4. Diagrama de bloques del sistema multiprocesador.

El sistema cuenta con dos chips los cuales poseen dos procesadores cada uno, con su respectiva memoria caché L1 y con otro nivel de memoria caché denotada como L2, la cual es accesible por ambos procesadores. Esta memoria L2 es compartida, por lo que es la misma para ambos chips. La comunicación entre los chips es creada a base de flags, al igual que la comunicación de los procesadores con las memorias caché y principal.

Diagrama de bloques del computador

Tomando en cuenta el diagrama anterior, el presente diagrama de bloques del computador modularmente hablando, es un nivel más alto, ya que se presenta la conexión de los chips del sistema con la memoria principal. A continuación se presenta la figura correspondiente a este diagrama.

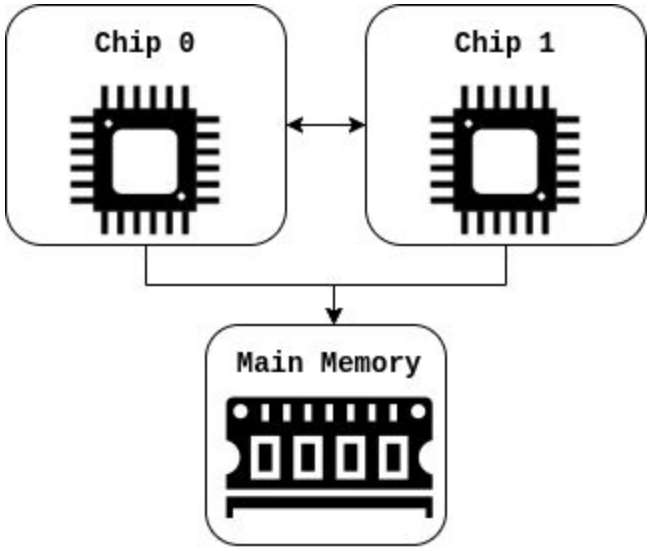

Figura 5. Diagrama de bloques del computador.

Cada chip contiene los bloques descritos en el diagrama anterior, y estos son conectado a una memoria principal. La comunicación de los chips con la memoria y entre ellos está realizada siguiendo el protocolo basado en monitoreo y directorios, utilizando al programar un conjunto de variables o flags, la cuales son utilizadas según el diagrama de estado del protocolo de coherencia y la política de escritura seleccionada.

Diagrama de flujo

Según los diagramas del protocolo de coherencia y la política de escritura escogida (Write-Through), se crea un diagrama de flujo, siguiendo paso a paso el funcionamiento del sistema. Dicho diagrama se presenta a continuación.

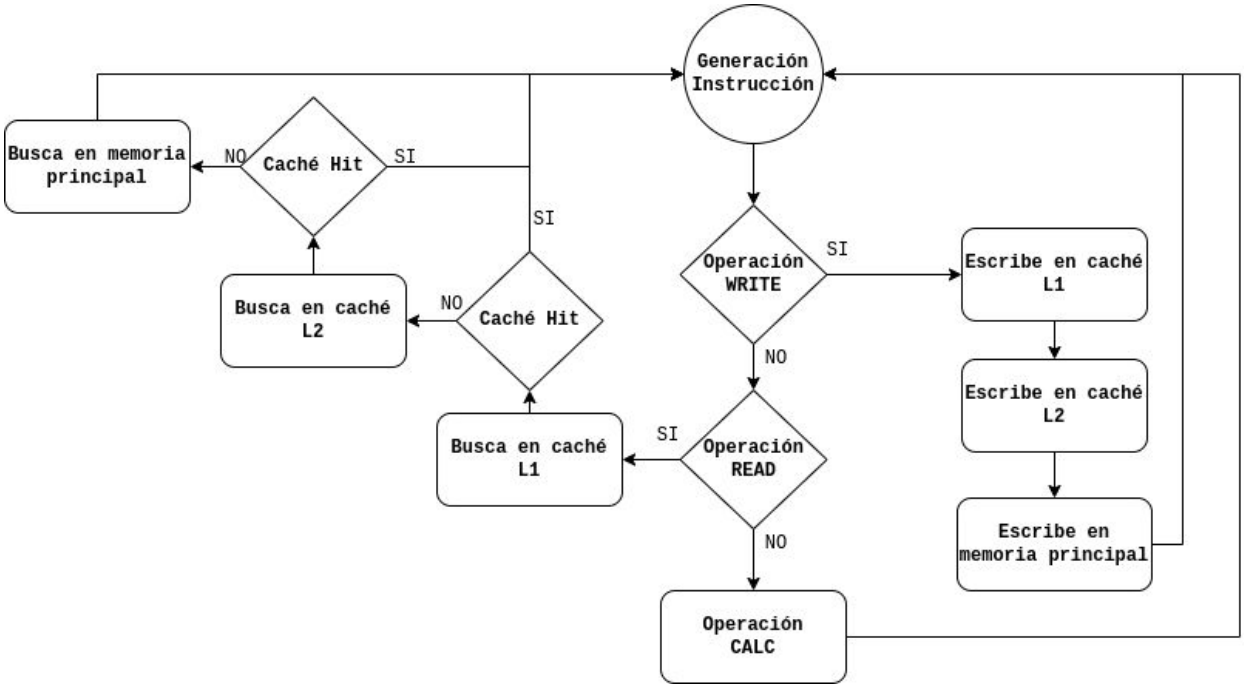

Figura 6. Diagrama de flujo del sistema.

Según el presente diagrama, se presenta el flujo de trabajo por procesador, teniendo en cuenta todos los componentes del sistema. Al ser un sistema multiprocesador, y según las especificaciones del proyecto, se tendrán cuatro procesadores realizando este flujo de manera simultánea.

Al crearse una nueva instrucción inicia el sistema a ejecutarse, verifica el contenido de la instrucción y si es de escritura, se escribe en todas las memorias, según la política de escritura establecida. Si la operación es de lectura, se lleva a cabo el flujo de buscar y leer el dato por todas las memorias, es decir, primero se busca el dato de L1, si se encuentra es leído y se termina el trabajo, pero si no, se repite el flujo pero en el siguiente nivel de caché (L2), y si no se encuentra, se termina yendo a buscar el dato a la memoria principal.

Diagrama de componentes

Para un mayor enfoque como desarrollo de software, se elabora un diagrama de componentes o módulos. Este tipo de diagrama muestra los módulos principales para la creación de cada uno de los bloques de la solución. A continuación, se muestra dicho diagrama.

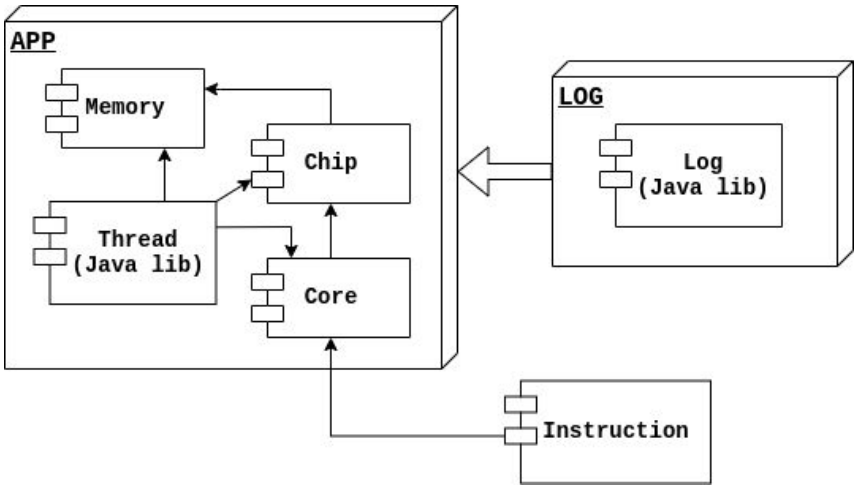

Figura 7. Diagrama de componentes del sistema.

En este punto es importante mencionar que los módulos de los cores, chips y la memoria dentro del sistema de la solución utilizan la biblioteca Thread de Java. Y para la creación de los archivos logs, los cuales poseen toda la información de los eventos generados por el programa, son implementados utilizando la biblioteca Log de Java.

Diagrama de clases

Teniendo en cuenta los paquetes a utilizar para el desarrollo del sistema, se crea también el diagrama de clases. A continuación, se muestra en la siguiente figura.

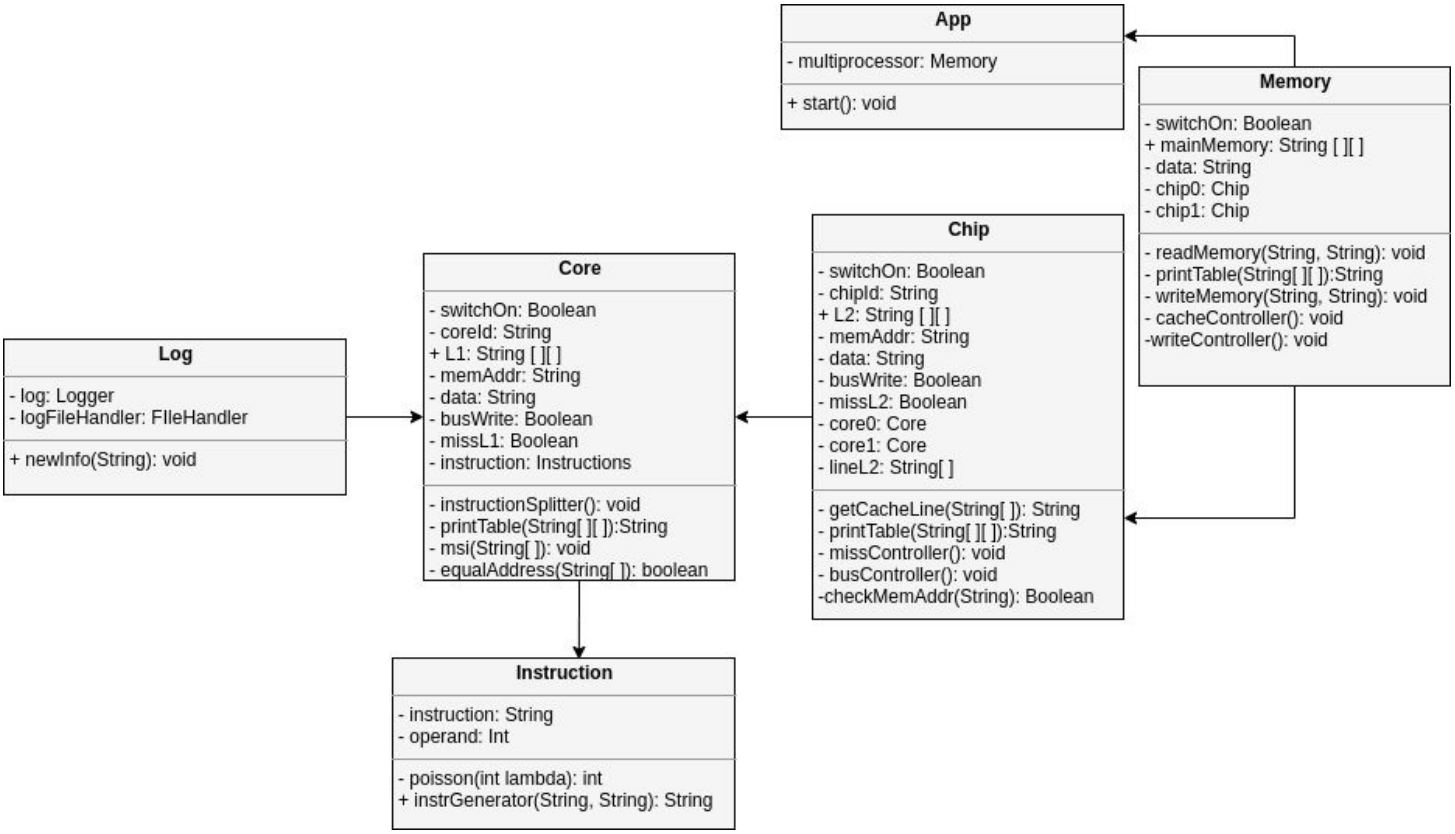

Figura 8. Diagrama de clases del sistema.

La clase Instruction es la encargada de generar mediante el método de poisson, las instrucciones aleatorias las cuales serán procesadas por el procesador, el cual viene siendo una instancia de la clase Core. La clase Core es la representación de los procesadores del sistema, esta se encarga de implementar el protocolo de coherencia MSI y controla a su vez la memoria caché L1 mediante el uso de flags.

La clase Chip es una abstracción de lo que el Cpu del sistema, este controla a los dos procesadores y a su vez la memoria caché L2 (lectura y escritura). Esta clase también posee la lógica que controla los misses que ocurran en el sistema mediante el uso de flags y se encarga del control de los datos del bus de comunicación con la memoria principal.

La clase memoria, además de dar control a la memoria principal, esta finaliza el sistema al dar con el método del controlador de caché, la cual finaliza el flujo de trabajo

al controlar los datos en se transmiten en el bus y leer o escribir datos en la memoria principal.

En este punto es importante mencionar que el control y flujo a través de las memorias es controlado mediante el uso de flags. Además, las memorias fueron simuladas utilizando matrices.

La clase Log es la encargada de crear los archivos .log e ingresar los eventos que ocurran a lo largo de la ejecución del sistema. Esta es de vital importancia ya que al no implementar interfaz gráfica, proporciona toda la información necesaria para validar el correcto funcionamiento del sistema.

Distribución de probabilidad formal

Para la generación de instrucciones del sistema se utiliza como distribución de probabilidad formal, la distribución de Poisson. Esta distribución es de las más importantes de variables discretas.

Dentro de las aplicaciones donde se utiliza esta distribución es en la modelización de situaciones en las que se necesita determinar el número de hechos de cierto tipo que se pueden producir en un intervalo de tiempo o espacio.

Dentro de la solución del proyecto, esta distribución es utilizada para generar el tipo de instrucción (READ, WRITE, CALC).

Referencias bibliográficas:

Hennesy, J. And Patterson, David. Computer Architecture: A Quan-titative Approach. 5th Edition. Elsevier – Morgan Kaufmann. 2012.

Distribución de Poisson. Recuperado de: https://www.uv.es/ceaces/base/modelos%20de%20probabilidad/poisson.htm

Dey, S. and Nair, M. Design and Implementation of a Simple Cache Simulator in Java to Investigate MESI and MOESI Coherency Protocol. 2014. Recuperado de: https://www.researchgate.net/profile/Somdip_Dey/publication/263004594_Design_and_Implementation_of_a_Simple_Cache_Simulator_in_Java_to_Investigate_MESI_and_MOESI_Coherency_Protocols/links/5405ebd70cf23d9765a79c4b/Design-and-Implementation-of-a-Simple-Cache-Simulator-in-Java-to-Investigate-MESI-and-MOESI-Coherency-Protocols.pdf