Proyecto I

María Alejandra Castrillo Muñoz ale.castrillom97@gmail.com Instituto Tecnológico de Costa Rica Programa de Licenciatura en Ingeniería en Computadores CE-4302 Arquitectura de Computadores II

***Resumen*—El proyecto consiste en una implementación en python de un sistema multiprocesador con los protocolos de coherencia de caché MSI y basado en directorios.**

***Keywords*—Coherencia, cache, protocolos, multiprocesadores, MSI, protocolo basado en directorios.**

I. Sistema desarrollado

El proyecto desarrollado consiste en un sistema multiprocesador que consta de una memoria principal y dos chips. Cada chip posee una memoria caché L2 un controlador y dos procesadores con una caché L1 cada uno como se muestra en la figura 1. Este proyecto fue desarrollado con python como lenguaje de programación y la herramienta Tkinter para interfaz gráfica.

El sistema se explicará por requerimientos, componentes y niveles para entender la interacción de los componentes pero principalmente las memorias y las acciones de los controladores.

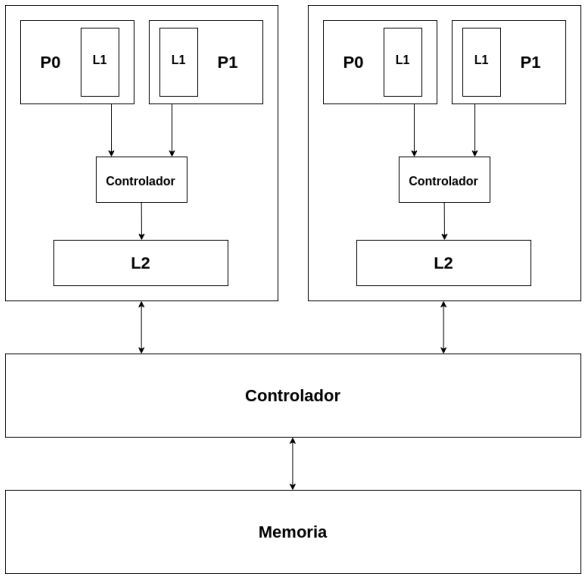

Figura 1. Diagrama de bloques del sistema.

Se puede dividir el sistema en dos niveles según el protocolo utilizado. El primer nivel lo componen los procesadores, las caché L1 el controlador del protocolo MSI y la caché L2 como se muestra en la figura 3. Se explicará el flujo y manejo de esta implemetación por medio los componentes de esta etapa empezando desde el de la izquierda hasta el de la derecha en la figura 3, ya que este sería el flujo de las instrucciones de accesos a memoria.

El nivel 2 lo componen la caché L2, el controlador del protocolo de directorios y la memoria principal.

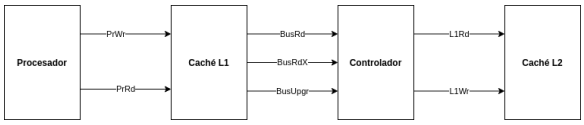

Figura 2. Diagrama de bloques del sistema.

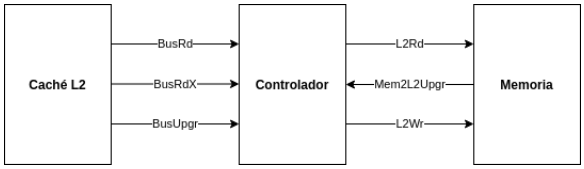

Figura 3. Flujo de transacciones para el protocolo basado en directorios.

Las transacciones utilizadas para el flujo dentro de los niveles corresponden a las siguientes:[1]

- PrRd: solicitud del procesador para lectura de una línea de caché.
- PrWr: solicitud del procesador para escritura de una línea de caché.
- BusRd: en caso de un read miss, se solicita al bus por el bloque de caché.
- BusRdX: en caso de que haya un write miss, se solicita al bus por el bloque de caché, e invalida el bloque en las demás caché.
- BusUpgr: en caso de un write hit, se le solicita el bus invalidar el bloque en las otras caché.
- Flush: el bloque de caché se ha escrito en memoria.

Se explicarán los componentes de cada nivel, así como su respuesta ante las transacciones ejecutadas.

I-A. Procesador

Al iniciar el procesador primero se deben generar las instrucciones que este ejecutará al correr el sistema.

I-A1. Generación de instrucciones: Se implementa una generación de instrucciones basada en una distribución formal binomial por medio de la biblioteca Numpy, en donde la posibilidad de que la instrucción sea de cálculo es de 0.4 la de una de lectura de 0.3 y la de una de escritura de 0.3, de manera que se evalúa en cada instrucción generada el tipo que se debe implementar, el resultado de las primeras instrucciones generadas se puede observar en la figura 5.

I-A2. Ejecución de instrucciones: Cada procesador cuenta con una función que se encarga de la ejecución continua de instrucciones y creación de nuevas en caso de haberlas ejecutado todas. El procesador toma una por una utilizando un orden FIFO y las ejecuta.

- CALC: en caso de ser una instrucción de cálculo se simula una espera de cierto tiempo asimilando el tiempo de ejecución de una de estas instrucciones.
- READ: el procesador debe leer el dato de la dirección de memoria proveída por la instrucción, para esto acude al nivel más próximo de memoria: L1. Por lo que hace una lectura: PrRd y espera por el resultado.
- WRITE: el procesador hace la lectura al nivel más cercano: L1. Para esto ejecuta la transacción PrWr.

I-B. Memoria caché L1

Esta memoria recibe peticiones del procesador y envía otras al controlador, a continuación las peticiones recibidas y su correspondiente ejecución (incluyendo las transacciones que genera a partir de estas). Las transacciones generadas se envían por medio de un bus y su respuesta se recibe por medio del mismo.

- PrRd: al recibir esta transacción la memoria L1 se encarga de verificar si contiene la dirección de memoria entre uno de sus bloques, en caso de tenerla verifica que el estado sea 'Só 'M'para devolverlo al procesador (Caché Hit). En caso contrario este se encarga de enviar una lectura al siguiente nivel por medio de la transacción BusRd y espera por su resultado.
- PrWr: en caso de lectura la memoria verifica que el bloque de caché no esté ocupado por un dato modificado, si no lo está, lo escribe, si sí lo está ejecuta la transacción BusRdX.En ambos casos ejecuta una transacción de BusUpgr para el dato que ha sido escrito.

I-C. Controlador de protocolo MSI

El controlador almacena todas las transacciones en una cola y las va ejecutando una por una por medio de un bus de datos, a continuación las transacciones recibidas y su correspondiente ejecución (incluyendo las transacciones que genera a partir de estas).

- BusRd: envía una lectura L2Rd a la caché L2 y espera por su respuesta.
- BusRdX: envía una escritura del dato a la caché L2.
- BusUpgr: actualiza el estado de la otra caché L1 que maneja esa misma dirección (si es que hay alguno) a Í'.

I-D. Caché L2

La memoria caché L2 cuenta con los estados 'DI', 'DM'y 'DS'. En esta además de tener un control por medio de estados, se tiene por medio de directorios, que en este caso implica dos espacios más en cada bloque de memoria que indican cuales memorias L1 de ese mismo procesador poseen el dato (Owners) y si hay otros chips haciendo uso del mismo (Shared Externally). A continuación las transacciones recibidas y su correspondiente ejecución (incluyendo las transacciones que genera a partir de estas).

- L1Rd: se verifica si se posee la dirección de memoria indicada dentro de esta caché. Si sí se verifica que su estado sea 'DMó 'DS', agrega el remitente a la lista de dueños del bloque y retorna el valor a la caché L1 remitente por medio del bus. En caso contrario se realiza un BusRd y se espera por el dato solicitado.
- L1Wr: se verifica si el bloque a ocupar se encuentra en estado 'DM', si no, se procede a almacenar el dato, el remitente le indica por medio de la petición si va a mantener el dato en caché, si sí entonces se agrega como dueño del bloque. Se realiza un BusUpgr para actualizar los bloques de memoria que posean esta dirección a 'DI'. Si, en caso contrario el bloque se encuentra en 'DMéntonces se realiza el mismo proceso pero además se hace una escritura a siguiente nivel, que en este caso corresponde a la memoria principal, con la transacción L2Wr.
- Mem2L2Upgr: es una petición de la memoria hacia L2 que indica que el bloque de memoria de la dirección indicada está siendo compartido por otro chip, por lo que se cambia el 'Shared Externallyá verdadero.

I-E. Controlador de protocolo de directorios

Este controlador aparte de la actualización de los estados, es encargado de la actualización de los dueños de cada bloque de memoria.

- BusRd: realiza la petición de lectura a memoria, retorna el resultado al chip y actualiza los dueños.
- BusRdX: Realiza la escritura a memoria y actualiza los estados del otro chip a inválido y los dueños.
- BusUpgr: actualiza el estado de la dirección dada en las otras caché a 'DI'.

I-F. Memoria Principal

La memoria principal se compone de 16 bloques y al igual que la caché L2 lleva control de los dueños que en ese caso serían los chips.

- L2Read: realiza la lectura a memoria, retorna el resultado al chip y actualiza los dueños, en caso de no tener el dato por haber sido modificado por alguna caché (DI) se solicita el dato al chip que lo tenga.
- L2Write: Realiza la escritura a memoria y actualiza los estados del otro chip a inválido y los dueños.
- MemUpgr: actualiza el estado de la dirección dada a 'DI'.

II. Resultados

En la figura 4 se puede ver la creación de todas las memorias y componentes al iniciar la ejecución del sistema.

En la figura 5 se pueden observar las instrucciones que se generan para un procesador al instanciarlo.

En la figura 6 se pueden observar las primeras ejecuciones luego de iniciar el programa, en donde se hace una lectura a la dirección de memoria 6 y hay una secuencia de misses hasta obtener el valor 'FFFF'de la memoria principal. También se da la escritura de la dirección 10 con el dato ÉA33'con un WRITE HIT y varias instrucciones de calculo.

En la figura 7 se puede ver una escritura a 6 posterior en donde se obtiene un write miss en L1 debido a que ya tenía un valor con el estado 'M'y se muestran las transacciones del bus hasta el Write hit en L2.

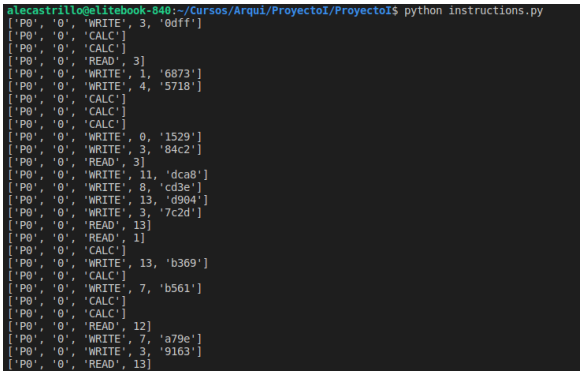

Figura 4. Inicialización de memorias.

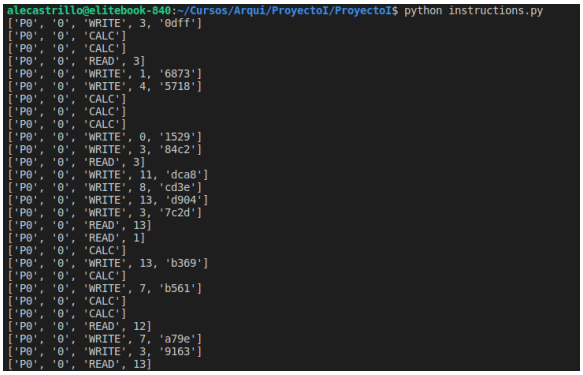

Figura 5. Instrucciones generadas.

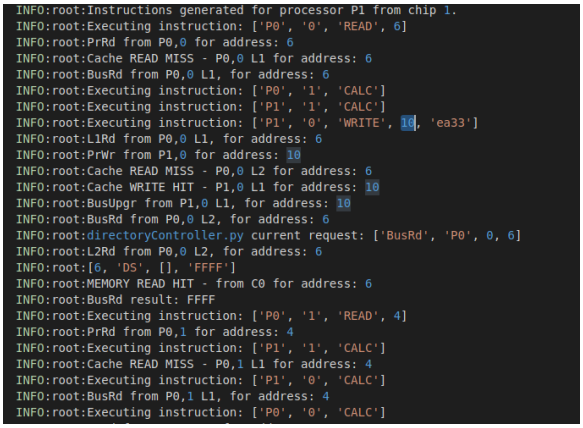

Figura 6. Secuencia de lectura hasta memoria después de la inicialización.

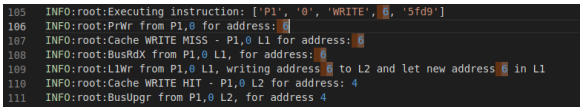

Figura 7. Secuencia de write con un write miss en L1.

III. Conclusiones

Los protocolos de coherencia de caché basados en estados aseguran la integridad de los datos y pueden proveer confiabilidad del sistema.

El protocolo MSI es ideal y fácil de implementar para memorias de primer nivel y el protocolo basado en directorios

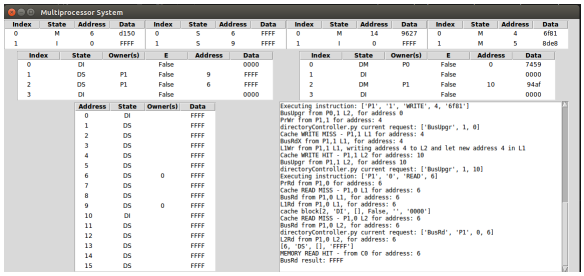

Figura 8. Visualización del sistema

provee un buen manejo de coherencia para niveles superiores de caché.

IV. Referencias Bibliográficas

Referencias

[1] J. Sánchez, D. Emer, "Lecture 7: Cache coherence," *Massachusetts Institute of Technology*, 2013. [Online]. Available: 10.1109/IPSN.2007.4379674