INSTITUTO TECNOLÓGICO DE COSTA RICA
ÁREA ACADÉMICA DE INGENIERÍA EN COMPUTADORES

CE4302: Arquitectura de Computadores 2

Proyecto I:
Simulador de Protocolo de Coherencia de Caché MESI

Estudiante:
Joshua Guzmán Quesada- 2018084240

Profesor:
Luis Barboza Artavia

Cartago, Costa Rica
Semestre II, 2022

1. Lista de Requerimientos del Sistema

-Proveer de una interfaz gráfica, donde se muestra mostrar en todo momento, los bloques de memoria de caché, memoria principal, bus principal y cuatro procesadores, así como de las instrucciones estado y dirección.

-Cada uno de los procesadores tiene una memoria caché L1, estos con cuatro bloques diferentes, los cuales son manejados con asociatividad one-way.

-Los bloques pertenecientes es la memoria caché deben poseer número, el estado según el protocolo, dirección y el valor del dato que estén manejando.

-El sistema debe emplear el sistema el protocolo de coherencia MESI para su control de memoria.

-La memoria principal del sistema es única y debe estar constituida de ocho celdas, comenzado desde 000 a 111 respectivamente.

-En lo que respecta a la memoria principal, los datos deben ser manejados de forma hexadecimal.

-El bus funge como el puente de comunicación entre lo que es la memoria principal y los bloques.

-El manejo de datos, escritura y lectura, así como la actualización de los mismos, deben manejarse por medio del protocolo MESI.

-La generación de las diferentes instrucciones se debe realizar por medio de alguna distribución probabilística, sin emplear el manejo de librerías.

-El sistema genera tres diferentes tipos de instrucciones, Write, Read y Calc.

-La simulación de la penalidad debe realizarse en lo que respecta a los accesos de memoria.

-La ejecución de los cuatro procesadores y las funciones asociadas a estas, deben realizarse por medio de hilos individuales.

-El sistema debe permitir ejecutar instrucciones paso a paso, donde se pueda ver como se ejecutan poco a poco cada una de estas.

-Inicializar el sistema de forma automática, lo que genera activamente las instrucciones y las muestra en pantalla.

-Detener todo el sistema de forma que no se ingresen instrucciones.

-Igual forma poder ingresar una instrucción, para esto se debe indicar el procesador destino, si es Read indicar la dirección si es Write indicar la dirección y su valor.

-El lenguaje de programación es libre.

2. Opciones de Solución al Sistema

Opción 1

La primera propuesta planea estar desplegada sobre C#, lo anterior responde a que, para poder desarrollar una interfaz gráfica acorde a los requerimientos solicitados, esto se puede lograr por medio de Visual Studio (Figura 1) el cual provee una mejor herramienta para la construcción de interfaz gráfica, aunque parezca un poco superficial, esto es importante, ya que es la interfaz uno de los pilares de este proyecto, si esta puede organizarse de una mejor forma todo estará ordenado. En lo que respecta al diseño lógico del proyecto, se planea moldear todo con clases y objetos, ya que al final se abstraerán elementos de hardware, por lo que el paradigma responde de buena forma a este modelo. En la misma línea se contempló utilizar la distribución de Poisson para poder generar las instrucciones, debido a que como solo se requería usar tres instrucciones, entonces su implementación no debía ser complicada.

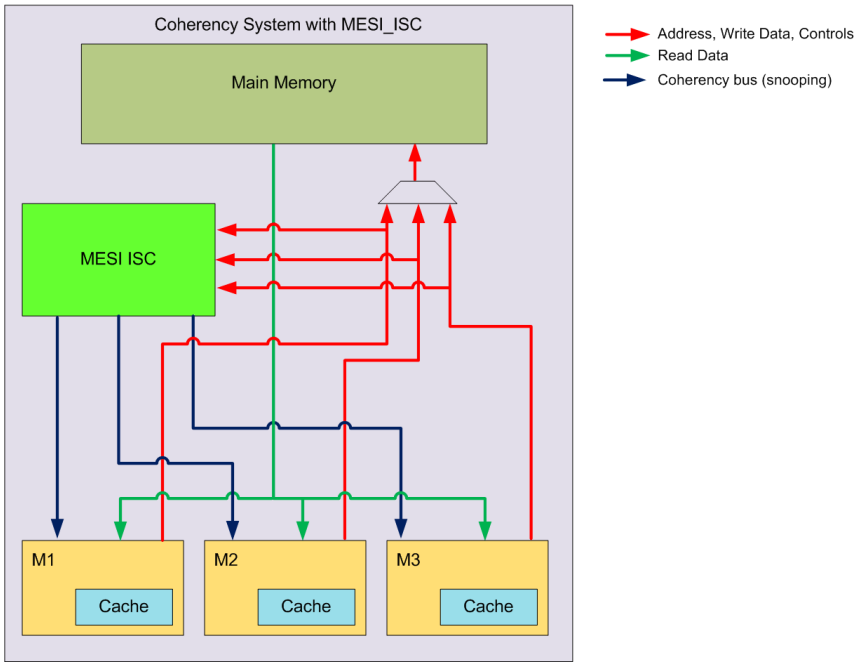

Figura 1. Modelado del diseño para el proyecto.

Opción 2

La segunda opción contempla el uso de Python y usar un modelado del paradigma funcional propio de este para poder ir conectando diferentes funciones en cadena, lo que permite ejecutar todo el sistema como un todo al final, de igual forma emplear el "tool" gráfico de Tkinter para poder desarrollar la interfaz hacia el usuario, pese a lo tedioso que puede resultar esta. En la sección lógica se quiere hacer uso de la distribución Binomial, debido al conocimiento previo que existe en esta debido al curdo de Probabilidad y Estadística, lo que permita un mejor desempeño de esta.

3. Comparación de Opciones de Solución

En ambas opciones de recalcar que la diferencia de tiempo es importante, ya que la amplia experiencia que existe en Python por sobre C# es muy considerable, pese a las facilidades en el diseño de interfaz de la segunda. Otro aspecto es poder moldear los objetos, debido a que el proyecto bien es un claro ejemplo de poder usar POO, en Python esto no es tan sencillo como en C#, por lo que hay que buscar la forma de solucionar esto, ya que el uso de funciones en cadena puede llegar a enciclar el programa seriamente o disminuir su desempeño. En lo que respecta a la distribución de Poisson o Binomial, el haber usado la segunda con anterioridad pues da la señal de poder ser empleada de mejor forma.

4. Solución de Propuesta Final

Entre las ambas propuestas expuestas anteriormente, considerando estas se lograr como solución la segunda opción, sin embargo, se tomarán los principios de programación orientada a objetos de la primera alternativa, ya que Python se adapta muy bien a esta, por lo que no hay problema alguno. El conocimiento previo sobre Python debido a los cursos anteriores es un pilar muy importante para tomar la segunda alternativa, ya que el tiempo y complejidad, variables muy importantes, se reducen considerablemente por lo que al final esta tiene las bases para el proyecto.

Link de YouTube

https://youtu.be/cPBMgSXJWY4

Link de Github

https://github.com/joshual777/jguzman_computer_architecture_2_2022.git