TRABAJO FINAL.

CREACIÓN E IMPLEMENTACIÓN DE LA MEMORIA CACHE.

Asignatura
Arquitectura de Computadoras

Alumnos
Lucas Suárez y Darel Urrutia

Universidad Nacional de la Patagonia San Juan Bosco

ÍNDICE

Descripción del Proyecto

El proyecto consiste en incorporar una memoria caché al simulador de la Máquina plus, que fue vista en la materia Elementos de Informática (EDI), donde realizamos nuestras primeras prácticas de programación en lenguaje Assembler.

La simulación de la Máquina plus se realizó en Logisim, un simulador de circuitos electrónicos que aprendimos a usar en la materia Arquitectura de Computadoras (AdC), correlativa a EDI.

El proyecto fue pasando por distintos estudiantes de la materia AdC, los cuales fueron agregando y mejorando las partes del circuito, como la memoria principal, la unidad de control, los registros, el uso de relojes, la tabla de instrucciones, etc.

Cache

Para incorporar la memoria caché, primero hay que entender cómo funciona, la función principal de una memoria caché es reducir el tiempo necesario para acceder a los datos almacenados, proporcionando un acceso más rápido que la memoria principal del sistema.

La memoria caché actúa como un buffer temporal para los datos que se utilizan con frecuencia. Cuando se accede a un dato por primera vez, se copia desde la memoria principal al caché. En los accesos posteriores, el procesador busca en la caché primero para ver si el dato ya está almacenado allí, y si lo encuentra, lo recupera desde la caché. Si el dato no está en la caché, se recupera de la memoria principal y se copia en la caché para su uso futuro.

Para el proyecto nosotros implementamos una caché de acceso directo que funciona mediante el uso de bloques de memoria. Cada bloque de memoria de la caché tiene una dirección de memoria asociada, y se almacena una copia de los datos que se encuentran en esa dirección de memoria en la caché. Cuando el procesador necesita acceder a un dato, busca en la caché utilizando la dirección de memoria del dato. Si el dato se encuentra en la caché, se recupera de forma rápida y se evita tener que acceder a la memoria principal, lo que reduce el tiempo de acceso. Si el dato no se encuentra en la caché, se tiene que buscar en la memoria principal y, si se encuentra, se carga en la caché.

Implementación de la caché en el proyecto

Para este proyecto utilizamos:

DIRECCIÓN DE MEMORIA: 16 bits para direccionar una RAM de 64K

CAMPO INDEX: 8 bits para direccionar la caché de 256 líneas.

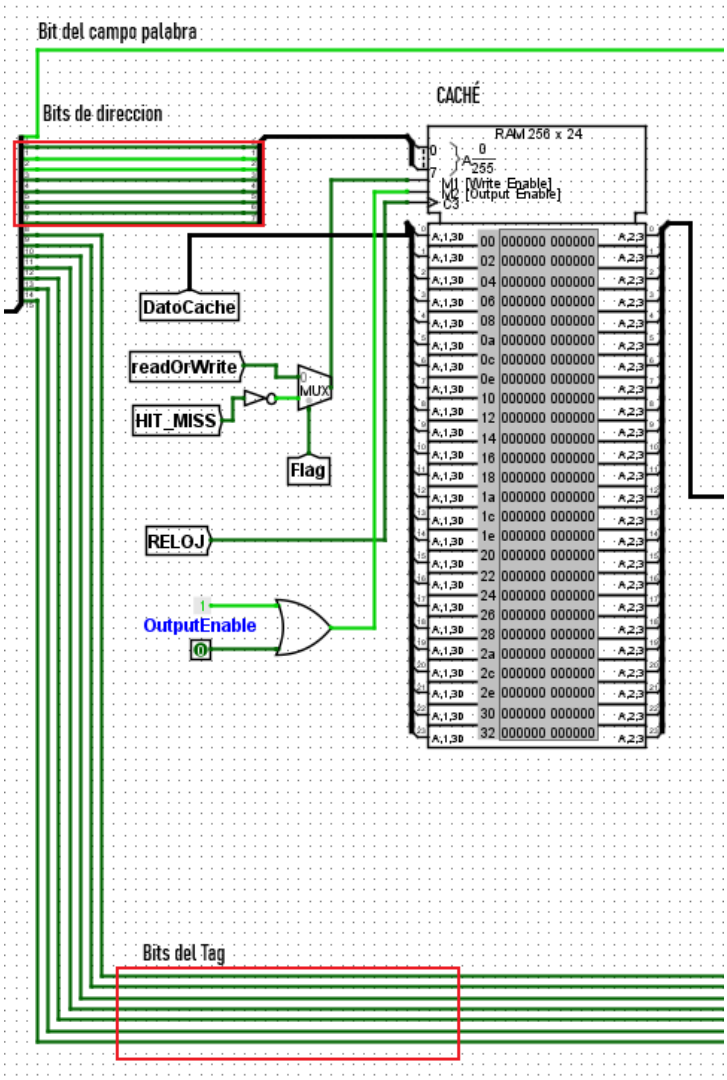
CAMPO PALABRA: 1 bit para seleccionar entre las 2 palabras de cada renglón.

CAMPO TAG: 7 bits. Resulta de: 16 - 8 - 1 = 7

Para el simulador creamos una caché del tamaño 256 x 24, el cual recibe 8 bits como direccionamiento de memoria y da como resultado 24 bits, los cuales los 16 bits menos significativos se dividen en 2 (8 bits para cada uno) esta parte del bloque de 24 bits son utilizados como la parte de la palabra, la cual en este caso elige cuál de las dos palabras usará, de los cuales salen el dato que tiene la cache.

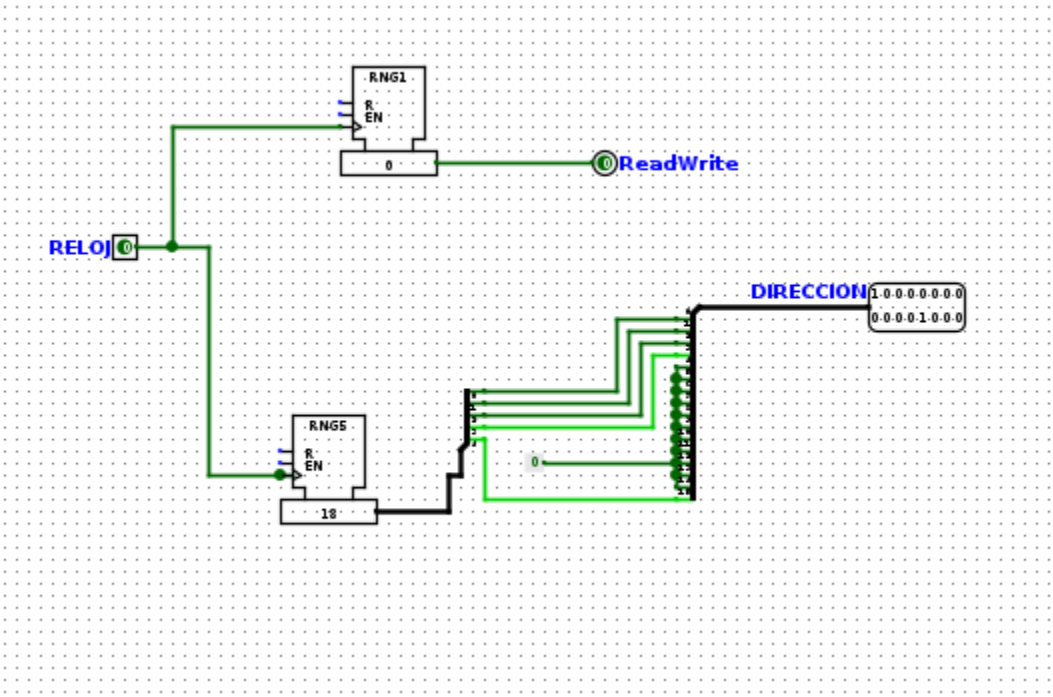
Luego los 8 bits que le siguen a los 16 bits anteriores son usados como el TAG del renglón de la caché.

Para el funcionamiento de la caché se utiliza la dirección de 16 bits esta es separada en tres secciones, el bit menos significativo es usado como el campo de palabra los próximos 7 son usados como direccionamiento, los cuales se usan para decir en qué posición de la caché estará guardado el dato, y los 8 bits más significativos son usados como el TAG, el cual se usará para comparar que el valor que está guardado en la caché en la posición elegida es correcto.



Para generar las direcciones de 16 bits, creamos un subcircuito que se encargue de esto, en este utilizamos un reloj el cual cada 3 ciclos genera una nueva dirección y que opción elige si este hará una lectura o una escritura. Estos datos son enviados al circuito general, el cual se encarga de las opciones elegidas.

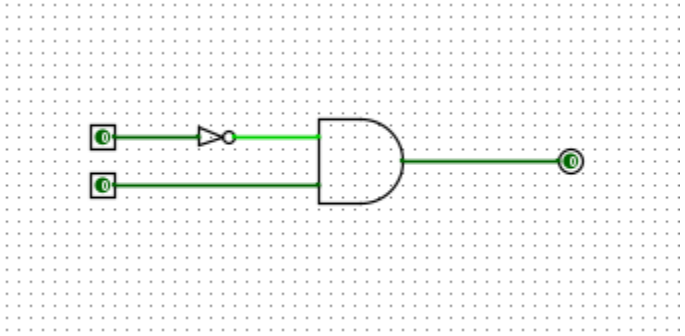
Subcircuito Generador de Direcciones



Cuando se genera la dirección de memoria esta es enviada a buscar en la caché el dato requerido, para comprobar que sea el dato correcto se realizan 2 comparaciones la primera será a través del bit de validez con el cual comprobaremos que el dato que está cargado en la caché es correcto, en caso de que el bit de validez no sea válido (el bit sea 0), tendrá que ir a la Memoria Principal a buscar este dato, en caso de que sea válido (el bit es 1), sabrá que el dato en la posición elegida esta actualizado, pero todavía no sabrá si el dato guardado es el requerido, para esto compara los dos Tag, el de la dirección y el del dato de la caché.

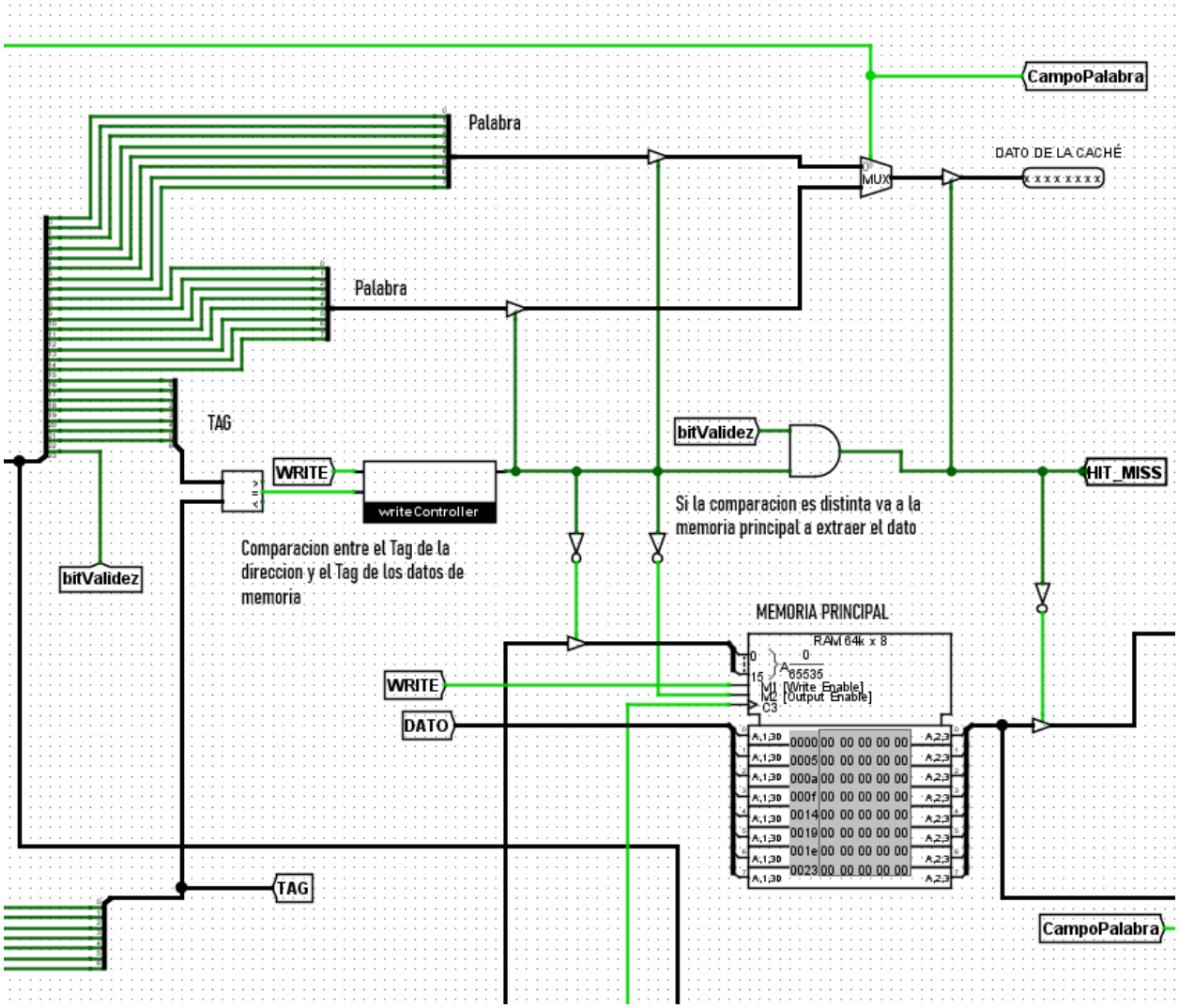
Antes de explicar las dos opciones hay que aclarar el funcionamiento del subcircuito writeControler.

Subcircuito writeController



Este subcircuito es sencillo, con este comprobamos que operación se está realizando, si la etiqueta WRITE está activa, este subcircuito no dejará que se extraiga el dato de la caché ya que se está efectuando una escritura la cual se tiene que realizar en la Memoria Principal. Pero si la etiqueta WRITE no está activa, significa que se está buscando un dato, el cual puede estar en la caché o en la memoria principal.

Volviendo a las comparaciones sucederán 2 posibles opciones:



Hay Hit

La primera será que los dos TAGS son correctos en este caso lo compara con el writeController el cual comprueba si está realizando una lectura o escrita, en nuestro caso estamos comprobando la lectura, por lo que en este caso la etiqueta WRITE estará deshabilitada, por lo cual, el writeController dará un bit válido, con este habilitaremos las dos posibles palabras que contiene la caché, con el bit del campoPalabra se selecciona cual de los palabras se obtendrá el dato.

No hay Hit

En este caso habrá dos posibilidades una es con el bit de validez, el cual ya explicamos anteriormente y el otro resultado será cuando el tag de la dirección y del dato de la caché no es el mismo, en ambos casos no se activaran las dos palabras de la caché y no se podrá obtener el dato de la caché, en este caso se tendrá que extraer de la Memoria Principal, para este se le pasa la dirección de 16 bits la cual será usada como dirección de memoria, con esta dirección se busca en la Memoria Principal y en esa dirección de MP se saca el dato y este sigue su camino en la Mplus, pero además de esto al mismo tiempo de extraer el dato, este es cargado en la caché.

Para esto creamos un controlador (Controller) el cual necesita los siguientes atributos:

El renglón de la caché donde se busco si estaba el dato.

El bit de la dirección que se usó para el campo palabra.

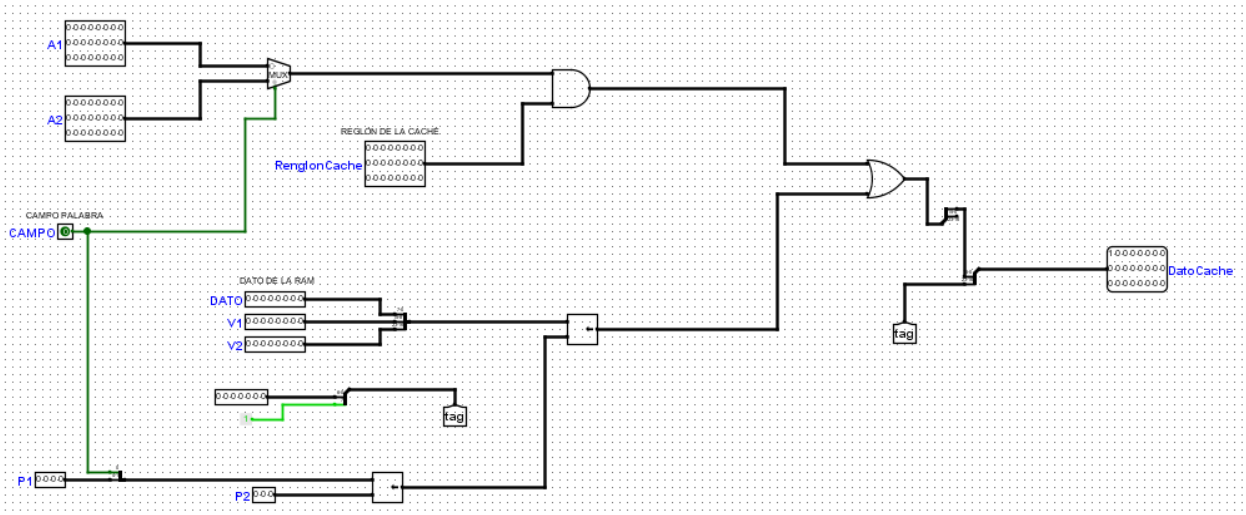
El dato extraído de la Memoria Principal.

El tag de la dirección.

Además de estos se le agregan otros más los cuales son atributos constantes.

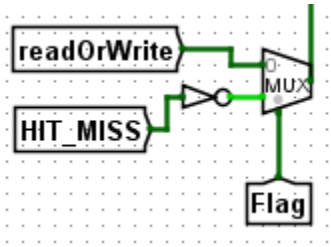
En este controlador se usan todos estos atributos con los cuales da como resultado el dato que será guardado en la caché, para guardarlo se habilita el bit de escritura en la caché y este es escrito en la posición dada.

Subcircuito “Controller”



El subcircuito “Controller” se encarga de actualizar el renglón de la caché, es decir, cuando se produce un miss o una operación de escritura esta sección del programa arma el nuevo renglón de la caché. En primera instancia, recibe el renglón de la caché, el dato extraído de la RAM y el bit del campo palabra. Con esta información, limpia, en el renglón, la sección donde iría la palabra. Simultáneamente, con el uso de shifter, desplaza el dato proveniente de la RAM para que puedan ser acoplados correctamente a través de una compuerta OR. Por último, una vez armado el renglón, actualiza el tag y carga el bit de validez.

Escritura y Lectura en Caché



Respecto a la escritura y lectura en la Memoria Caché, esta sección del programa se ocupa de activar el modo de escritura o lectura en la caché. Su funcionamiento consiste en recibir el bit que indica si la CPU necesita hacer una operación de lectura o escritura, y, por otro lado, en recibir el bit que nos señala si hubo un hit o miss. Estos dos bits son multiplexados con el uso de

un flag que se activa cada 2 tick de nivel alto y bajo (El tiempo necesario para que el sistema realice el requerimiento de la CPU). Por Ejemplo, si nos llega una tarea de lectura de la CPU, en primer lugar, el bit “readOrWrite” será 0 y el flag también, por lo que se leerá en la caché. Posteriormente, en caso de que se produzca un fallo, el flag ya estará levantado y, gracias a la compuerta NOT, el bit de hit o miss estará activo. Entonces, se activará el modo escritura en la caché para que se pueda introducir el dato de la RAM.

Mejoras

Cambiar el tipo de cache, la cache de acceso directo es la forma más simple es fácil de implementar y tiene una baja latencia de acceso a los datos almacenados en la caché. Pero, puede haber muchos conflictos de caché si hay una gran cantidad de bloques de memoria que se asignan a la misma ubicación en la caché.

Por esto se puede utilizar la caché asociativa por conjuntos. Este tipo de caché es más complejo que la caché directa, pero tiene una menor cantidad de conflictos de caché. También tiene una mayor capacidad y flexibilidad, lo que permite manejar grandes cantidades de datos.

La caché directa es más adecuada para sistemas simples con pocos datos y una tasa de acceso relativamente baja. La caché asociativa por conjuntos es más adecuada para sistemas con una gran cantidad de datos y una alta tasa de acceso. Sin embargo, la implementación de una caché asociativa por conjuntos es más compleja y, por lo tanto, puede tener un mayor costo.

Mejorar la política de reemplazo de la caché, una mejoras pueden ser FIFO (primero en entrar, primero en salir) y LRU (menos recientemente utilizado). Si la política de reemplazo se ajusta para maximizar la probabilidad de que los datos relevantes se mantengan en la caché, puede mejorar significativamente el rendimiento de la caché.

Aumentar el tamaño de la caché: A medida que se aumenta el tamaño de la caché, se reducen las posibilidades de que se produzcan colisiones en la memoria caché. Esto aumenta la probabilidad de que los datos requeridos se encuentren en la caché, lo que mejora el rendimiento.

Conclusiones

Luego de realizar este trabajo llegamos a la conclusión de que es una buena forma de reforzar los temas vistos en la asignatura, fue una experiencia interesante y desafiante. Nos pareció interesante la evolución que fue teniendo este proyecto de la Máquina plus hecho por varios estudiantes, y como este puede ser utilizado de manera didáctica y que todavía tenga la posibilidad de mejorarse en futuros proyectos. También brinda una comprensión sobre el

funcionamiento de un procesador a los estudiantes y que de esta manera puedan ver cómo funciona la Máquina plus internamente e incluso hacer pruebas. Además, este proyecto nos permitió adquirir habilidades valiosas y comprender mejor el funcionamiento interno de los sistemas informáticos, además el proyecto nos brindó la oportunidad de profundizar en el conocimiento de los sistemas de memoria.

Referencias

1. Las distintas documentaciones que fueron realizadas por cada estudiante que trabajó sobre el circuito de la Máquina plus y material dado por las cátedra de Arquitectura de Computadoras.
2. Logisim: (http://www.cburch.com/logisim/docs/2.1.0-es/guide/)
3. Caché directa: (https://ajbeas.webcindario.com/directa.html)
4. Memoria Caché: (https://www.fdi.ucm.es/profesor/jjruz/WEB2/Temas/EC6.pdf)