Universidad de Costa Rica

Escuela de Ciencias de la Computación e Informática

CI-0120 Arquitectura de computadoras

Examen 1

Entregables 1, 2, 3

Profesor:

Francisco Arroyo Mora

Estudiante:

Josef Ruzicka González B87095

8 / 10 / 2021

Entregable 1.

Descripción y componentes de la cache.

Como la cache tendrá 8 lineas de 4 bytes cada una, tendremos una cache de 32 bytes. Cada linea de la memoria tendrá sus bits divididos en 3 campos: Un “tag”, un “set number” y un “line offset”.

Set number corresponde al número de set al que se desea acceder, como tenemos 8 lineas en nuestra memoria cache, y es 2-way associative la cantidad de sets que tendremos serán 8/2 = 4. Necesitamos 2 bits para identificar estos 4 sets.

Tag (Etiqueta) es utilizado una vez que ya conocemos en cuál set de la memoria cache debemos buscar para identificar cuál linea presente en el set es la que se debe elegir. Para esto utilizaremos los restantes 14 bits.

Line offset se utiliza para identificar cuál byte específico se desea acceder en una linea de la memoria cache. Como cada linea tiene 4 bytes, necesitamos 2 bits para identificar a cuál de estos bytes específicos debemos acceder.

Condición de miss.

Un miss ocurrirá cuando se ingrese una dirección para buscar en la memoria cache, una vez que se haya encontrado el set correcto, se procederá a buscar el tag de la línea solicitada, si ningúna linea en el set presenta el tag correcto significa que el bloque de información que se está buscando no ha sido cargado en la memoria cache desde la memoria principal.

Componentes de lineas de cache.

Cada línea de la memoria cache será parte de un set compuesto por un par de líneas. A nivel interno tendrá una dirección que se refiere al bloque de la memoria principal que se encuentra cargado en la linea de memoria cache.

Entregable 2.

Se construyó un circuito en logisim que se utilizará para representar cada una de las lineas de la memoria cache. El circuito recibe un Tag que corresponde a los 12 bits que se utilizarán para comparar si el dato que el CPU está buscando se encuentra almacenado el la memoria de la memoria cache identificada como “Line Data”. En caso de que el comparator utilizado para la verificación detecte un mis, se almacenará el bloque de memoria buscado desde la memoria principal del programa, en este circuito está etiquetada como “Data-in”.

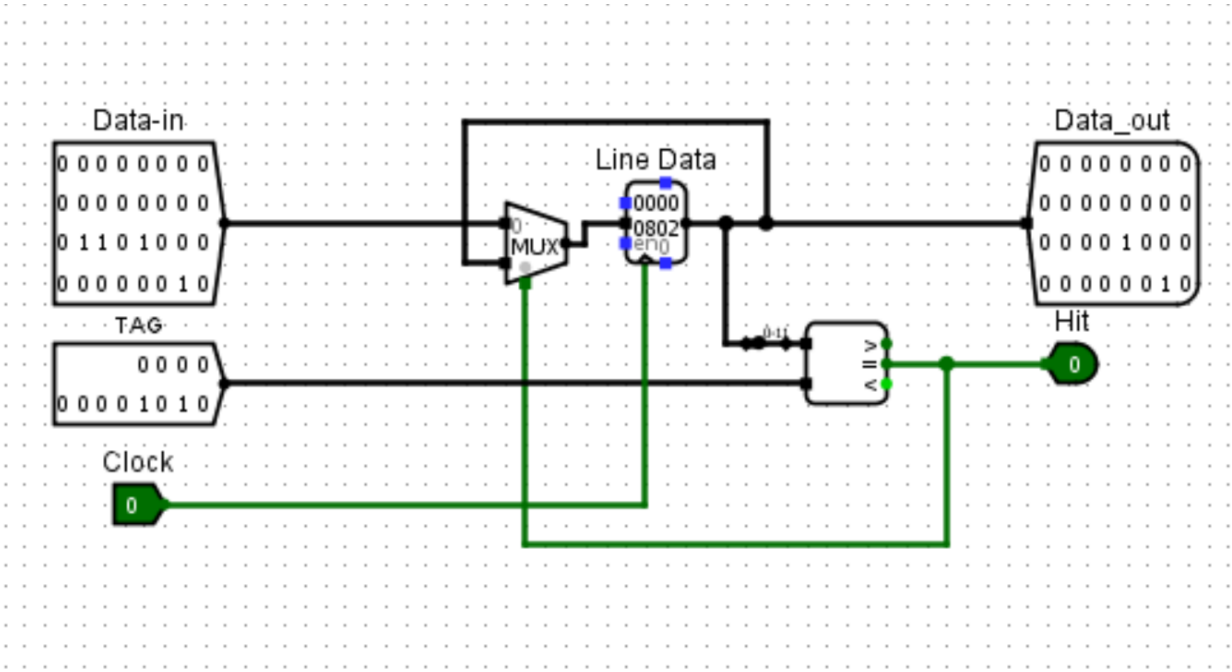

Imagen #1 CacheLine.circ

Entregable 3.

Para probar el funcionamiento de las lineas de cache se diseño en Logisim un circuito principal que presente una memoria cache identificada como “Cache” donde se pueden observar las lineas separadas por pares correspondientes a sus respectivos sets a quienes se accederá utilizando el demux que utiliza los 2 bits de set del Address que corresponden a Set, una memoria principal ROM identificada como “Main Memory” y una serie de multiplexores y demultiplexores que se encargarán de ubicar la linea de cache deseada por el Address de entrada. La etiqueta Data_out se utiliza para poder

observar el contenido del bloque de memoria referenciado, este número se envía al demux para separarlo en 4 bytes y un mux para que los bits de Offset del Address se utilicen para seleccionar el byte deseado del bloque de memoria. En cada set se utilizará el valor de la salida “Hit” para determinar cuál de las dos lineas es la que se desea utilizar.

La parte del circuito que se refiere a Datapath corresponde al Address, al Data_out y el Byte. Lo que corresponde a estructura de control se refiere al Clock.

Al hacer este circuito se encontró una limitante para escoger la linea correcta del set, ya que una vez que se accede al set ambas lineas se comparan con el Tag del Address por lo que si alguna línea ya tenía el Tag correcto, es posible que la otra linea copie el bloque de memoria desde la memoria principal también.

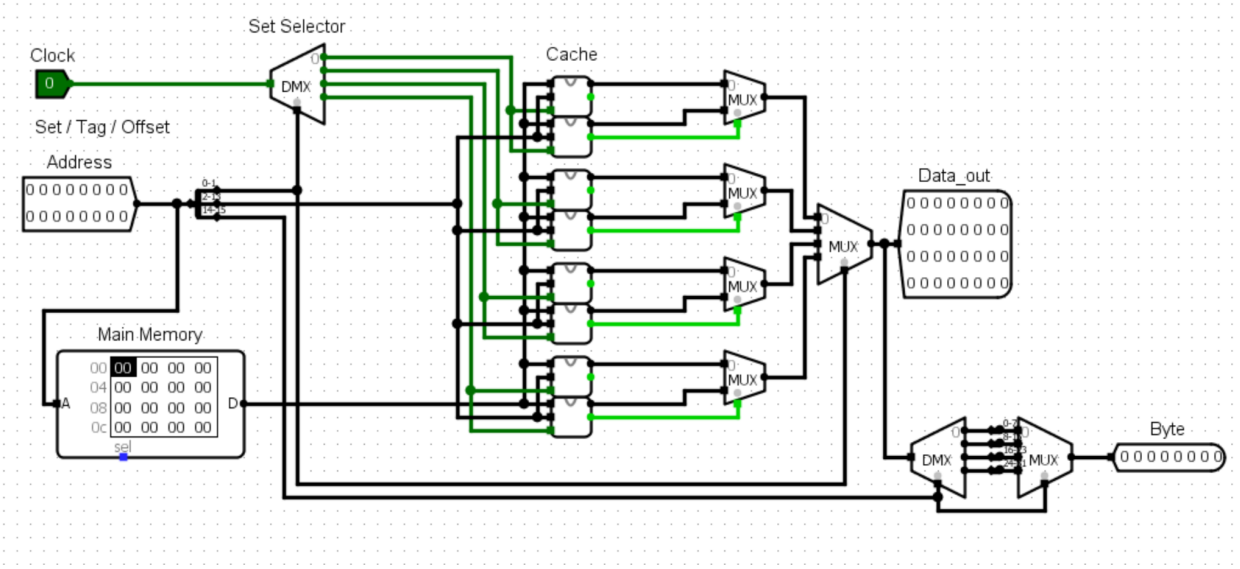

Imagen #2: Cache.circ

Referencias

John L. Hennessy y David A. Patterson, Computer Architecture: A Quantitative Approach. Elsevier Inc. / Morgan Kaufmann Publishers, San Francisco, 6ª edición (2019) [Cap. 2 y Apend. B] William Stallings, Computer Organization and Architecture, Pearson, 11ª edición (2019) [Cap. V]

https://courses.cs.washington.edu/courses/cse378/09au/lectures/cse378au09-19.pdf

https://www.d.umn.edu/~gshute/arch/cache-addressing.xhtml