Grayson Colwell and Jessica Cashman

Design Document

**Direct Mapped:**//how to calculate cache index, tag, word offset & how to place in cache

**Two-way Associative:**

//how to calculate cache index, tag, word offset, and how to implement the LRU algorithm

**Fully Associative:**

When looking at the implementation of the word offset in the fully associative implementation, the same implementation for the previous two methods was used. The address passed into the function was “anded” with 0x00000003 to get the lower two bits of the address as the word offset. Because this method was fully associative, it meant that the cache locations could hold the address passed into the function. Each of the 8 cache addresses were defined as being the tag at each cache address shifted left two bits and added with the word offset. Based off these addresses, each cache location was checked to see if the address was found in any of the locations. The tag was the remaining bits after calculating the word offset, so the address passed in shifted right two bits would contain the tag for each word in the cache.

When implementing the LRU algorithm, the first thing done was to create two temporary variables. These two variables used were tempLast and tempLastIndex. The cache was then parsed through to find which of the locations was used the least and based off this, this is where the address block from mainMemory would be placed in the cache..