**Iplc\_sim\_init:**

This function takes an index, the block size, and the associativity of the cache as parameters. Before allocating the cache it will perform checks to make sure the cache meets pre-established specifications such as ensuring the cache does not go over the max cache size. It uses this information to allocate memory and dynamically create the cache with the proper associativity.

**Iplc\_sim\_trap\_address:**

This function deals with checking if a given address is in the cache. It is called by various pipeline functions and considers the given associativity in addition to searches through the cache data structure. It will update the counter for hits, misses, and cache accesses. After looking through the appropriate entries for the address it will call the appropriate function to deal with a hit or a miss.

**Destroy\_cache:**

Goes through each element of the cache and deallocates the memory.

**iplc\_sim\_LRU\_replace\_on\_miss:**

When it is determined that an element is not in the cache, this function is called. It takes a cache index and a tag as parameters, both of which are determined by the caller. It puts the element in the cache and declares it as the most recently used entry. It then updates the entry that was previously the most recently used entry. If the cache is full, the LRU item is then replaced.

**iplc\_sim\_LRU\_update\_on\_hit:**

This function takes in a cache index and a given associative entry. It is called if an element has already been determined to be in the cache and it will update the element’s information. It will set it as the MRU and update the previous MRU information on the cache.

**Pipeline functions**

**Main**

**Iplc\_sim\_init:**

* Allocate cache
* Perform checks
* Set associativity

**iplc\_sim\_LRU\_replace\_on\_miss:**

* If cache is full remove LRU element
* Insert new element
* Set new element as MRU and update MRU info in cache

**iplc\_sim\_LRU\_update\_on\_hit:**

* Update element information
* Set element as MRU and update the previous MRU

**Iplc\_sim\_trap\_address:**

* Check if address is in cache
* Update access and miss counters
* Call appropriate function