**SET ASSOCIATIVE CACHE**

**Implementation and Analysis of Operations**

**Function Descriptions:**

* **getN():**

**Input:** starting point , requested block (Block Size)

**Output:** Starting Address

From the starting point till the requested block checking whether the bit is 1 if its 1 its returning the starting point (which is now updated i.e. The address)

* **createMask():**

**Input:** length, offset

**Output:** return 32 bit Address

It creates number of Length bits ON and number of Offset bits OFF and join them together, so it will become 32 bit Address and then return that address.

* **memAccess():**

**Input:** CacheLine Object1 (already saved Tag, Index, Offset), CurrentAccess(Input), CacheLine Object2(empty, Passed By refference)

**Output:** assign Object2, Prints CurrentAccess.

Break CurrentAccess address into Tag, Index and Offset and assign those values to object2.

* **cacheSetIsValid():**

**Input:** Cache, Associativity value, requested block

**Output:** Boolean, prints Valid bit

Goes to the set in the cache using requested block's index, and then iteratively looks through the whole set. If it finds a block with valid bit on, it prints "Valid bit: 1" and returns true. Otherwise it prints "Valid bit: 0" and returns false.

* **cacheSetTagMatch():**

**Input:** Cache, Associativity value, requested block

**Output:** Boolean, prints Tag bit (valid or not)

Goes to the set in the cache using requested block's index, and then iteratively looks through the whole set. If it finds a block with the same tag as the requested block , it prints "Tag match bit: 1" and returns true. Otherwise it prints "Tag match bit: 0" and returns false.

* **insertIntoCache**():

**Input:** Cache, Associtivity of cache , CacheLine Object1

**Output:** No output, block is either inserted or replaced.

Using index, goes to the corresponding set in cache and then sequentially scans the whole set. In the case of valid bit not set, it is set to valid. Set the tag as the requested block tag and print insert. If cache is already full then replace some value randomly (replacement index), set its tag as the requested block tag and print replace.