END-SEM ASSIGNMENT



Name-Lakshita Mathur

Section-A

Roll Number-2019170

INDEX

1. Question
2. Basic idea-

* Direct Mapping
* Fully Associative Mapping
* Set Associative Mapping

1. Code

* Direct Mapping
* Fully Associative Mapping
* Set Associative Mapping

1. Input
2. Output

**1)Question**

**Cache Assignment**

A cache of size S with CL as the number of cache lines and block size B is to be built. S, CL, and B are in powers of 2. Write a program that allows loading into cache and searching cache using:

1. Direct mapping
2. Associative memory
3. n-way set associative memory where n is a power of 2.

Any programming language (of your choice) can be used.

**2.1) Direct Mapping**

The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. or  
In Direct mapping, assignee each memory block to a specific line in the cache. If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed. An address space is split into two parts index field and a tag field. The cache is used to store the tag field whereas the rest is stored in the main memory. Direct mapping`s performance is directly proportional to the Hit ratio.

i = j modulo m

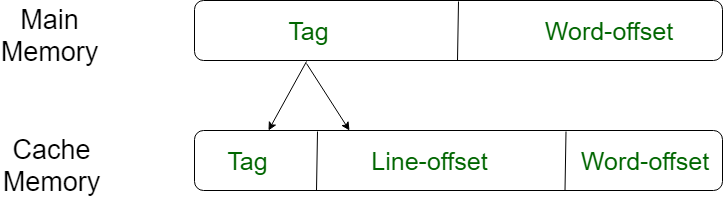
where

i=cache line number

j= main memory block number

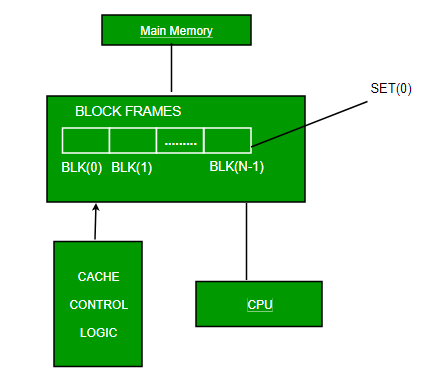
m=number of lines in the cache

For purposes of cache access, each main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory. In most contemporary machines, the address is at the byte level. The remaining s bits specify one of the 2s blocks of main memory. The cache logic interprets these s bits as a tag of s-r bits (most significant portion) and a line field of r bits. This latter field identifies one of the m=2r lines of the cache



**2.2) Fully Associative**

In this type of mapping, the associative memory is used to store content and addresses of the memory word. Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form.



**2.3) N-way Set Associative**

This form of mapping is an enhanced form of direct mapping where the drawbacks of direct mapping are removed. Set associative addresses the problem of possible thrashing in the direct mapping method. It does this by saying that instead of having exactly one line that a block can map to in the cache, we will group a few lines together creating a ***set***. Then a block in memory can map to any one of the lines of a specific set. Set-associative mapping allows that each word that is present in the cache can have two or more words in the main memory for the same index address. Set associative cache mapping combines the best of direct and associative cache mapping techniques.

In this case, the cache consists of a number of sets, each of which consists of a number of lines. The relationships are

m = v \* k

i= j mod v

where

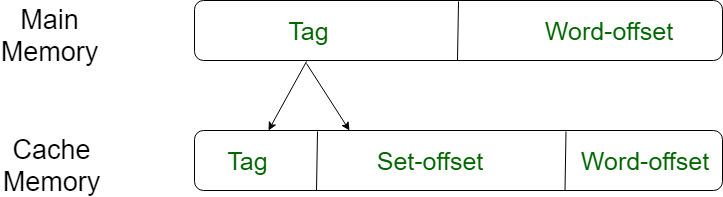
i=cache set number

j=main memory block number

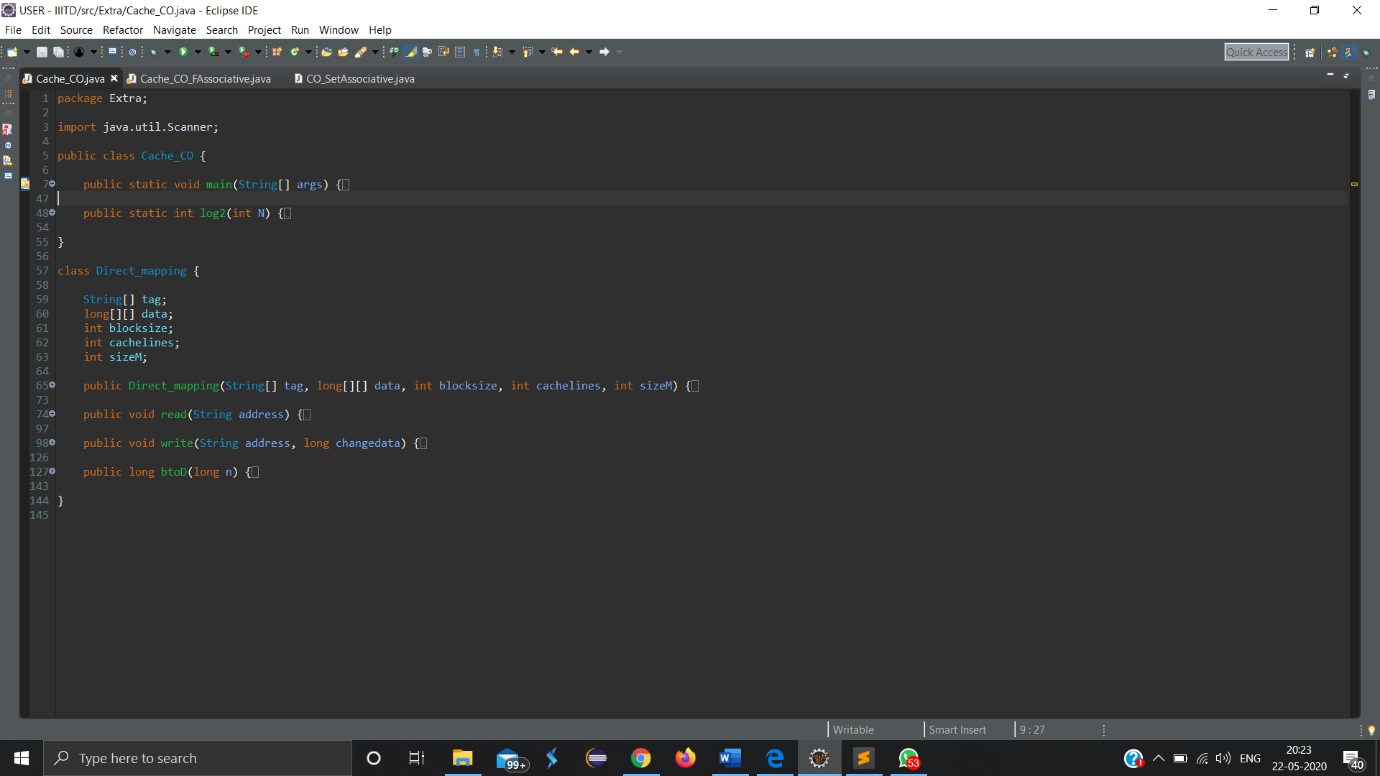
v=number of sets

m=number of lines in the cache number of sets

k=number of lines in each set



**3.1) Direct Mapping Code**



Direct Mapping class contains two main functions i.e., read and write and one helper function i.e., btoD.

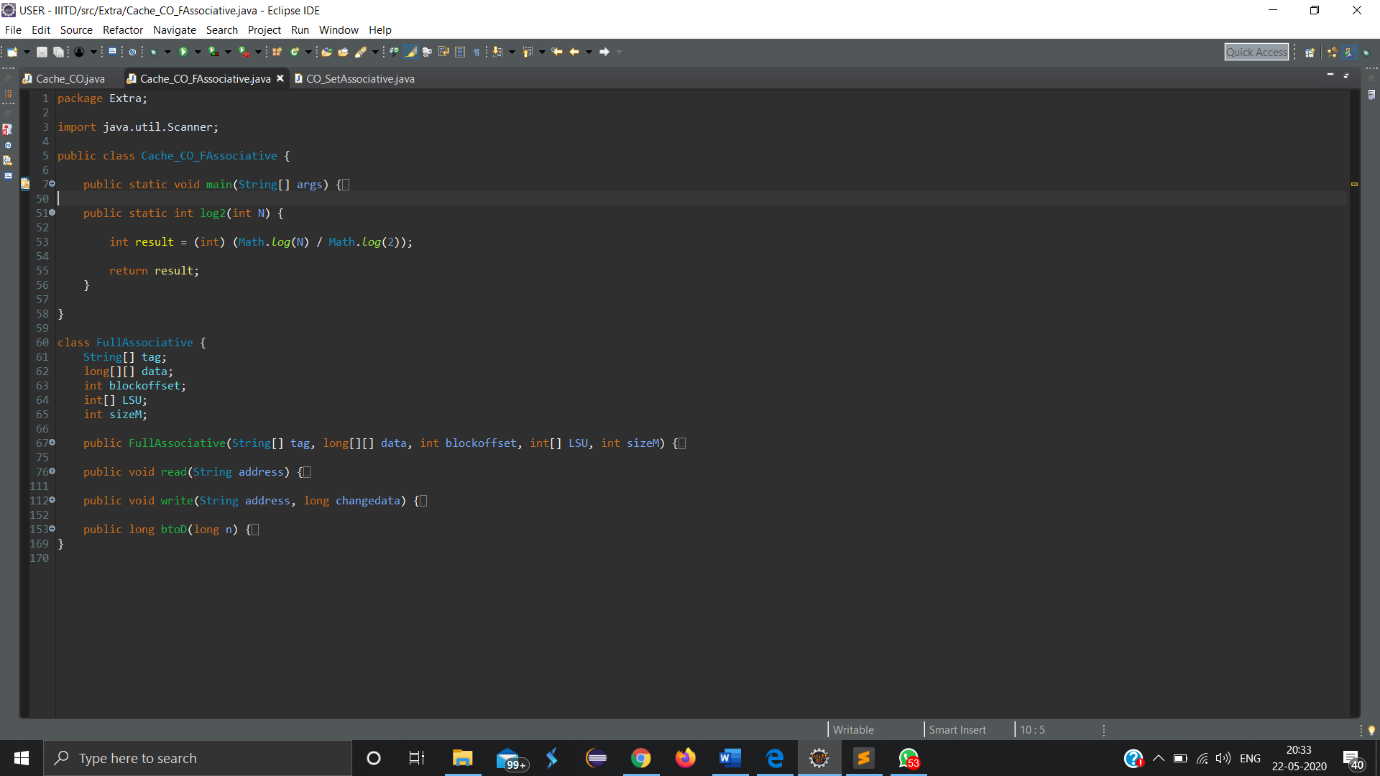
**Cache Hit**

A cache hit describes the situation where your site’s content is successfully served from the cache. The tags are searched in the memory rapidly, and when the data is found and read, it’s considered as a cache hit.

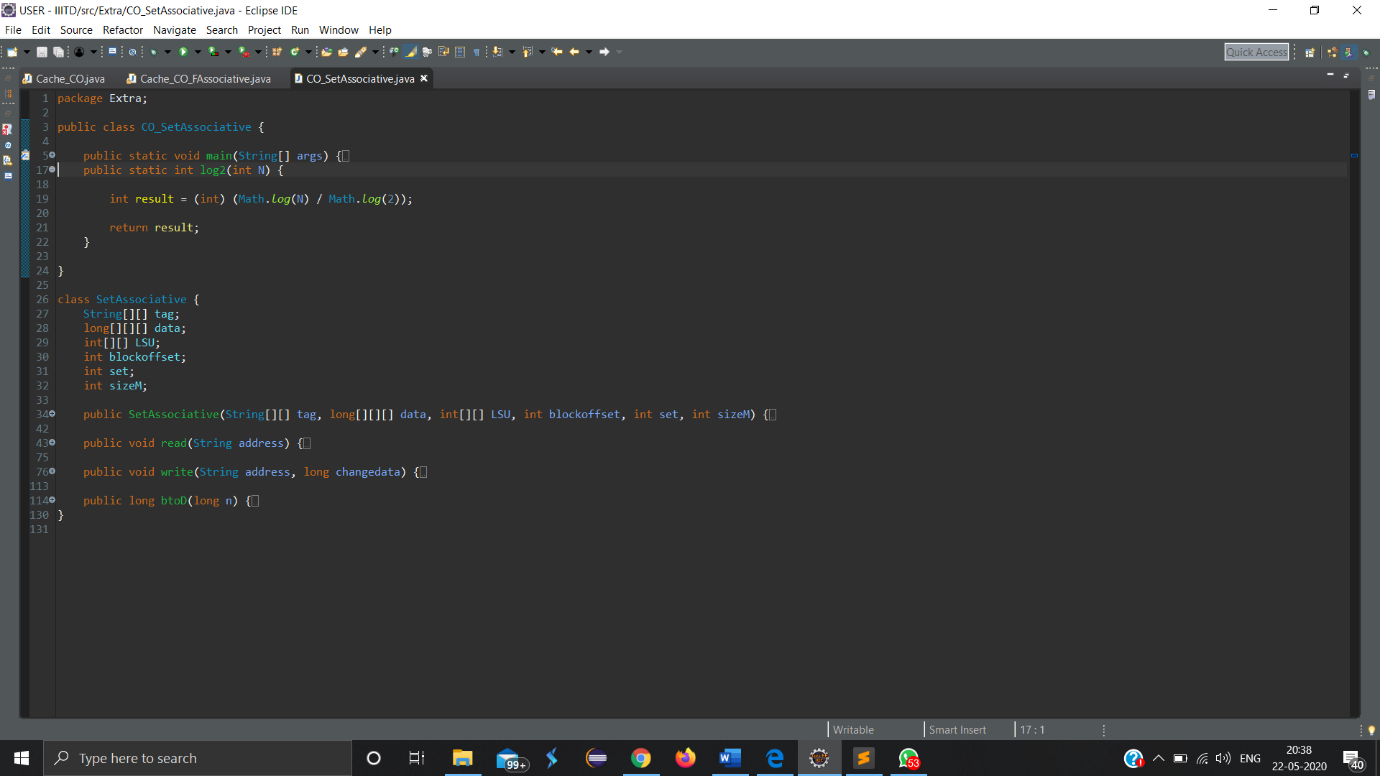
**Cache Miss**

A cache miss refers to the instance when the cache is searched and the data isn’t found. When this happens, the content is transferred and written into the cache. As we are not maintaining a main memory so when we replace the address and data it is replaced by -1.

**3.2) Fully Associative Mapping Code**



**3.3) Set Associative Mapping Code**



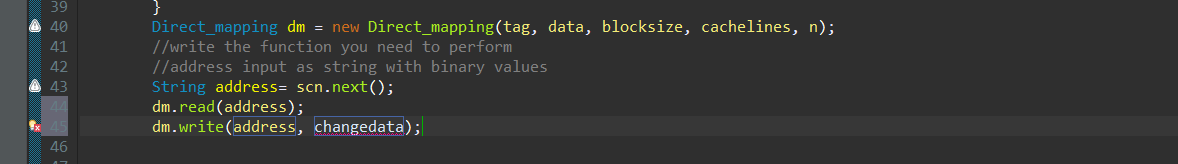
**4) Input**

First line contains Integer n which represents number of bits and n raise to power represents words in main memory. Second lines contain Integer blocksize which represents number of bits of block offset and 2 raises to power blocksize represents number of words in one line of cache. Third line contain Integer cachelines which represents number of bits of line number and 2 raises to power cachelines represents number of cache lines.

**Direct Mapping**

We can further take inputs for tag array in string and for data array in long.

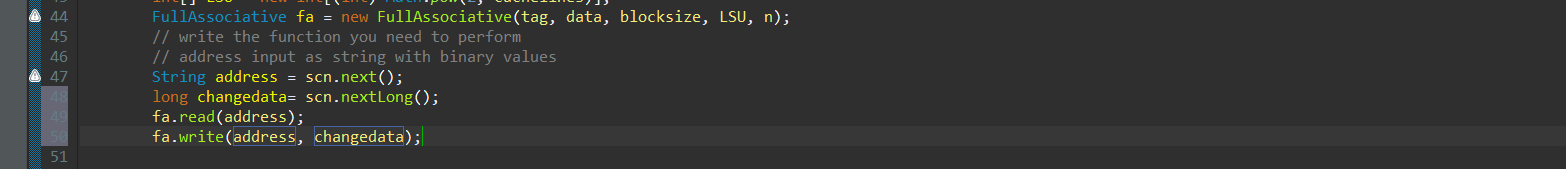
For running read or write function we take address input as string. We can run them as:



**Fully Associative Mapping**

We can further take inputs for tag array in string and for data array in long. And LSU array which act as a counter value telling the least and most used tag value.

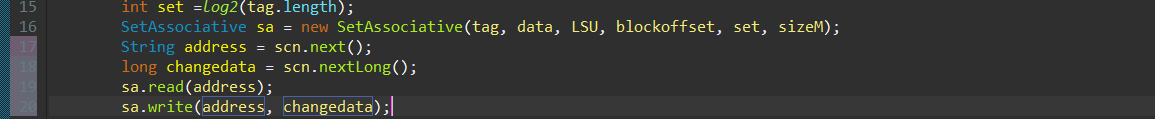
For running read or write function we take address input as string. We can run them as:



**Set Associative Mapping**

We can further take inputs for tag array in string and for data array in long. And LSU array which act as a counter value telling the least and most used tag value.

For running read or write function we take address input as string. We can run them as



**6)Output**

In output we have print telling if we hit cache or we miss cache.

**Read**

If we hit cache, data at the given address in printed.

If we miss the cache, we replace the address in the cache and print the data.

**Write**

If we hit cache, we update the new data at the given address.

If we miss the cache, we replace the address in cache and update the new data at the given address.