DOCUMENTATION
CACHE PROJECT

Cache is a small fast memory that stores data which has most recently been accessed. This cache project is able to show cache operations like loading the cache using mapping, searching and writing data to it.

I have initialized a machine word length to 32 bits but can be changed if required. All inputs are taken as integers(including memory address). Later these inputs are converted to binary for checking the bits of various elements required for mapping purpose.
Number of cache lines and block size are taken as input followed by type of mapping the user wants the data to be mapped as. Once the data gets mapped, read or write operations can be performed keeping the type of mapping same.

Some important variables declared:
c_rows=no. Of rows in the cache(cache lines)
c_cols=no. Of columns in the cache(block size)
cache_array[][]=contents of cache memory
tag_set[],tag_bits[],tag_array[]=keeps track of the tags for different mappings
associate_counter=keeps track of the cache filled in associate mapping
set_counter[]=keeps track of cache lines filled in each set
num_set=number of sets
lines_in_each_set=Value of n (in case of n-way set or set size)
num_hits=number of hits;
num_miss=number of misses;

Direct mapping: Maps each block of main memory into only one possible cache line. Its like one-one Mapping. There is a block_offset,line_num and tag which basically represent the no. Of bits of- the block size, cache lines, the most significant bits(machine word length-line_num-block_offset) respectively. All of them are converted to integers to be used as array indicies. The line_num is used to denote where the memory has to placed(in which line no,). Block_offset gives the specific position inside a block to be accessed in the cache.

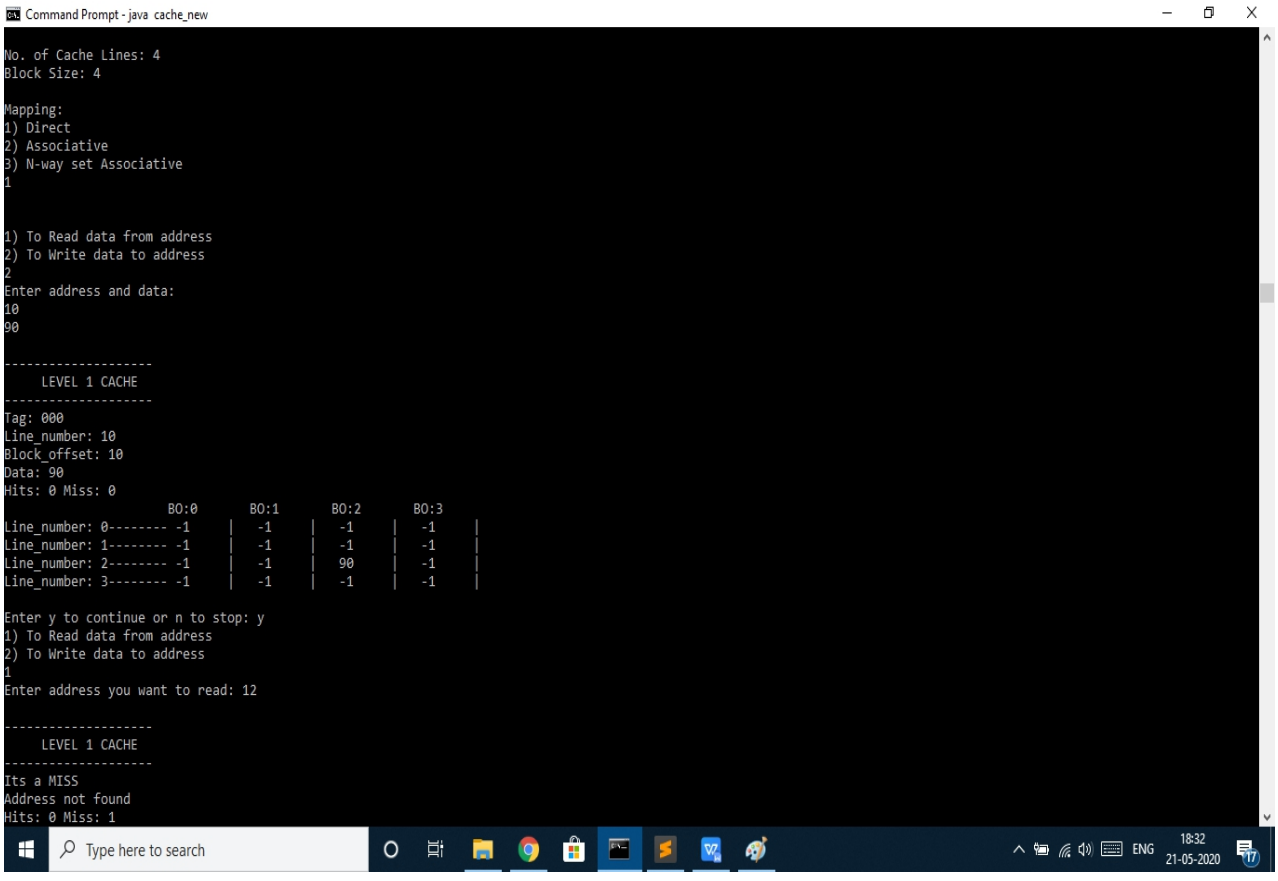
Associative mapping: Any block of the main memory can be placed anywhere in the cache. Since there is rule to place the block thus in this case the there is a block_offset and tag(tag+block_offset=machine word length). In this case searching takes time as each cache line needs to be checked for the tag value.

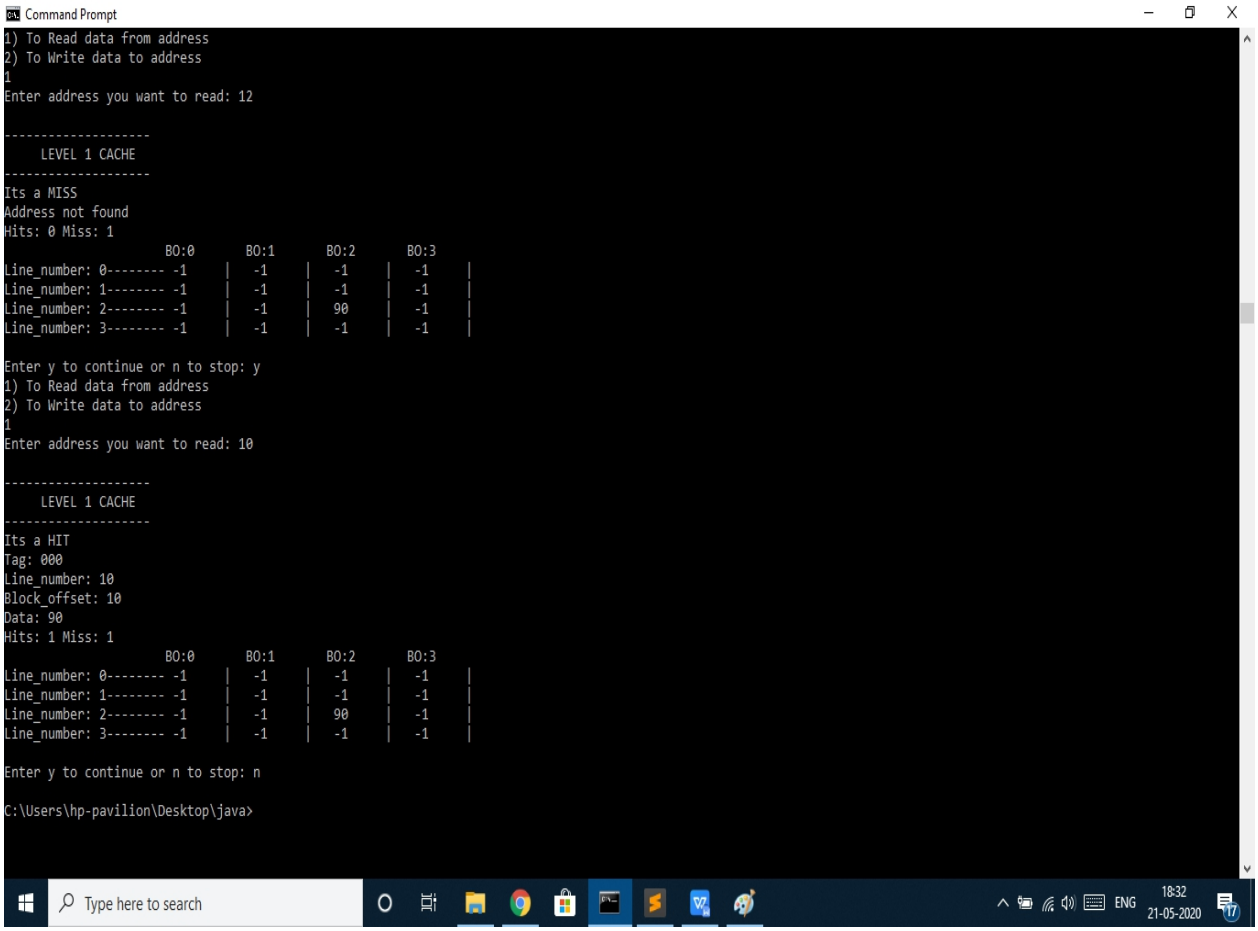
N-**set Associative:** In this case the cache lines are divided into sets of n. Although mapping to a set is direct but mapping inside the set is taken as associative.In this case as well a tag along with a set_number,where set_number is the no. Of bits(no. Of bits of no. Of sets) before the block_offset. The set_number2 decides the to

which set the block will go. Then the block will be placed inside the set via association(with the help of set_counter).

Replacement is done in case the tag does not match(tag is the main identifier of a block placed in cache). Although for direct-map replacement is built-in which is controlled by the (virtual or physical) memory address while for the rest the criteria followed for the same is FIFO principle.Count of hit number and misses is also maintained. Initially the cache is empty which is denoted by “-1” in all memory address. After every read or write operation the current cache is displayed. It is important to note that read and write is done directly to cache.

This is a screenshot of my cache project. It is an example of direct mapping with read and write operations.“-1” represents no data in the cell of the block. BO is Block_offset.
The machine word length is 32 bits. The No. Of cache lines is 4 and Block Size is 4.




Command Prompt
1) To Read data from address
2) To Write data to address
1
Enter address you want to read: 12
LEVEL 1 CACHE
Its a MISS
Address not found
Hits: 0 Miss: 1
BO:0 BO:1 BO:2 BO:3
Line_number: 0-------- -1 | -1 | -1 | -1 |
Line_number: 1-------- -1 | -1 | -1 | -1 |
Line_number: 2-------- -1 | -1 | 90 | -1 |
Line_number: 3-------- -1 | -1 | -1 | -1 |
Enter y to continue or n to stop: y
1) To Read data from address
2) To Write data to address
1
Enter address you want to read: 10
LEVEL 1 CACHE
Its a HIT
Tag: 000
Line_number: 10
Block_offset: 10
Data: 90
Hits: 1 Miss: 1
BO:0 BO:1 BO:2 BO:3
Line_number: 0-------- -1 | -1 | -1 | -1 |
Line_number: 1-------- -1 | -1 | -1 | -1 |
Line_number: 2-------- -1 | -1 | 90 | -1 |
Line_number: 3-------- -1 | -1 | -1 | -1 |
Enter y to continue or n to stop: n
C:\Users\hp-pavilion\Desktop\java>
Type here to search
ENG
18:32
21-05-2020