COMPUTER ORGANISATION
CSE-112
CACHE MEMORY ASSIGNMENT

Problem Statement:
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.
Language used: Python

Logic:
A main memory array after taking Memory size and Block size as inputs from the user is created. The main memory contains X blocks where X=(Memory Size/Block Size) along with their respective addresses.

```
Input Memory size128
Input Block size8
Number of bits for block indeces= 4.0
Input Cache Size32
ENTER K FOR K-WAY ASSOCIATIVE MAPPING2
NUMBER OF SETS FORMED= 2.0
NUMBER OF BITS REQUIRED FOR REPRESENTING SETS= 1.0
Number of cache lines= 4.0
Number of bits for cache line indeces: 2.0
4

        MAIN MENORY

0000  B0
0001  B1
0010  B2
0011  B3
0100  B4
0101  B5
0110  B6
0111  B7
1000  B8
1001  B9
1010  B10
1011  B11
1100  B12
1101  B13
1110  B14
1111  B15
```

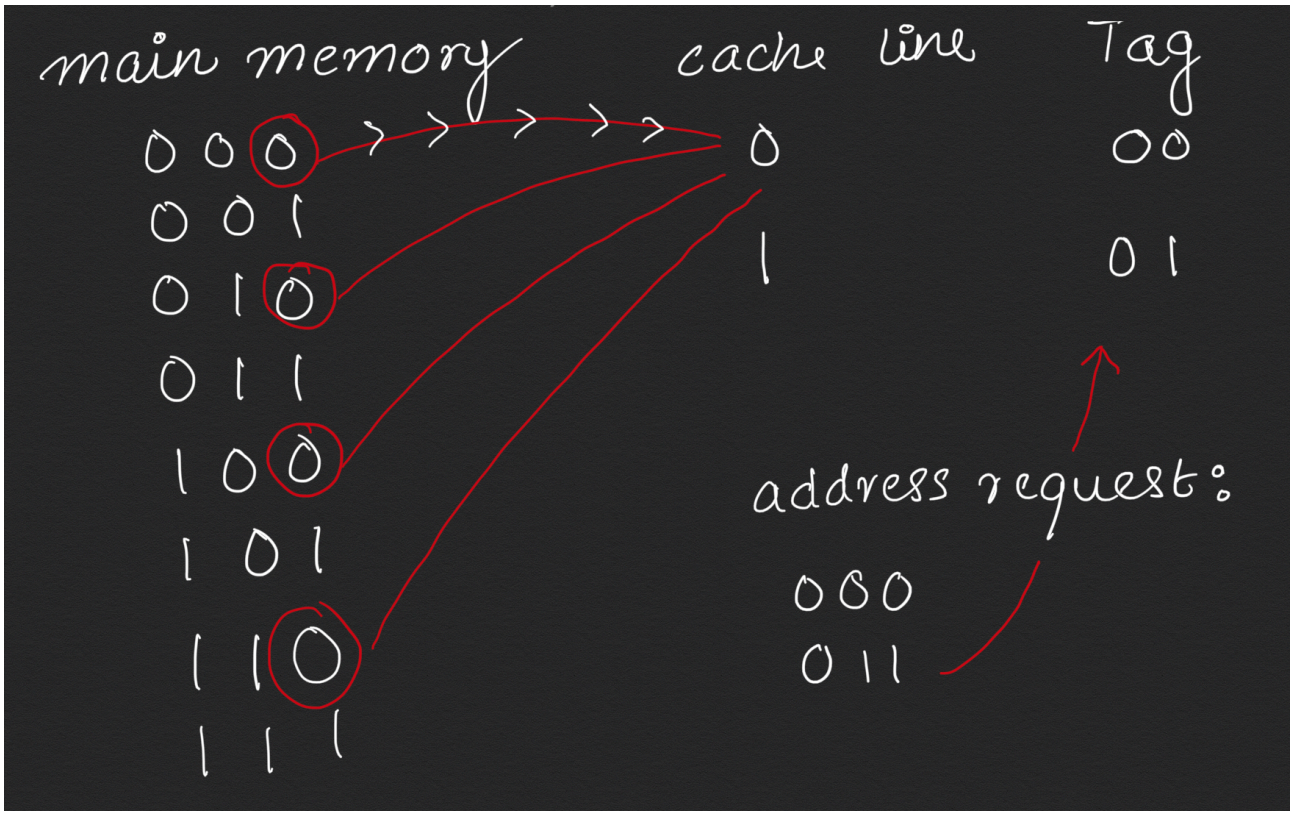
addtocache():
This function is used to copy memory blocks from Main memory to cache memory. In this, the address which is taken as input is broken down into segments: Set, Tag, Line.
These segments are used in respective mapping techniques in order to add the address to cache.
This function also takes care of situations when the cache is full, and when address is not present in main memory.

search():
This function is used to search for a block address in cache. In this, the address which is taken as input is broken down into : Block Offset, Tag and Cache Line/ Cache set. The cache line/set is used to determine the position of block in the cache memory and the tag bits are used to verify the presence of the memory block in cache. The block set tells that which word in the particular block is requested by the CPU. The search function prints “CACHE HIT” if the block requested by the CPU is found in the cache and prints “CACHE MISS” if the block isn’t found. In case of a cache miss, The memory block is searched for in the main memory and then added to the cache.

Approach used for given mapping techniques:

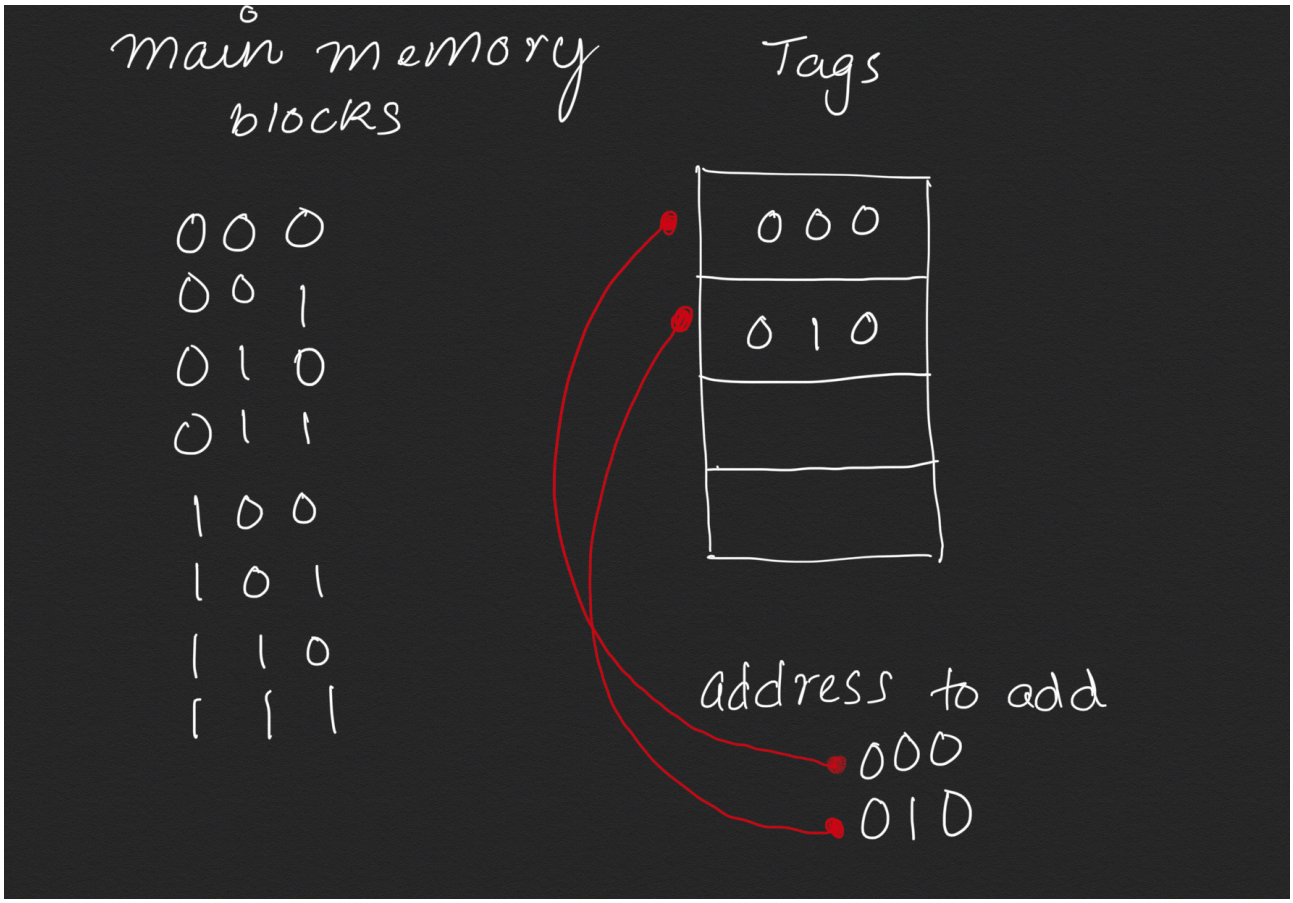
1) DIRECT MAPPING:

In this case, each memory block is associated to a cache line in the cache based on the LSB(s) of the block address. The address that needs to be added to the cache is broken down into Tag and Cache Line. An array consisting of possible cache lines is traversed and a search is made to make sure that the required cache line exists in the cache. If the cache line is found at some index in the array, the Tag is added to the tag array at the index corresponding to the cache line. While searching, the address in the request Made by the CPU is broken down into Tag, Cache Line and Block offset. The Cache line is searched in the cache, if presence is confirmed at some index and the tag is also presents at the same index in the tag array, it's a cache hit. In case of a cache miss, the address requested by the CPU is added to the particular cache line and Tag array. If there is already a block present in the cache associated to the same cache line, the block is simply replaced with the new block who's address was requested.



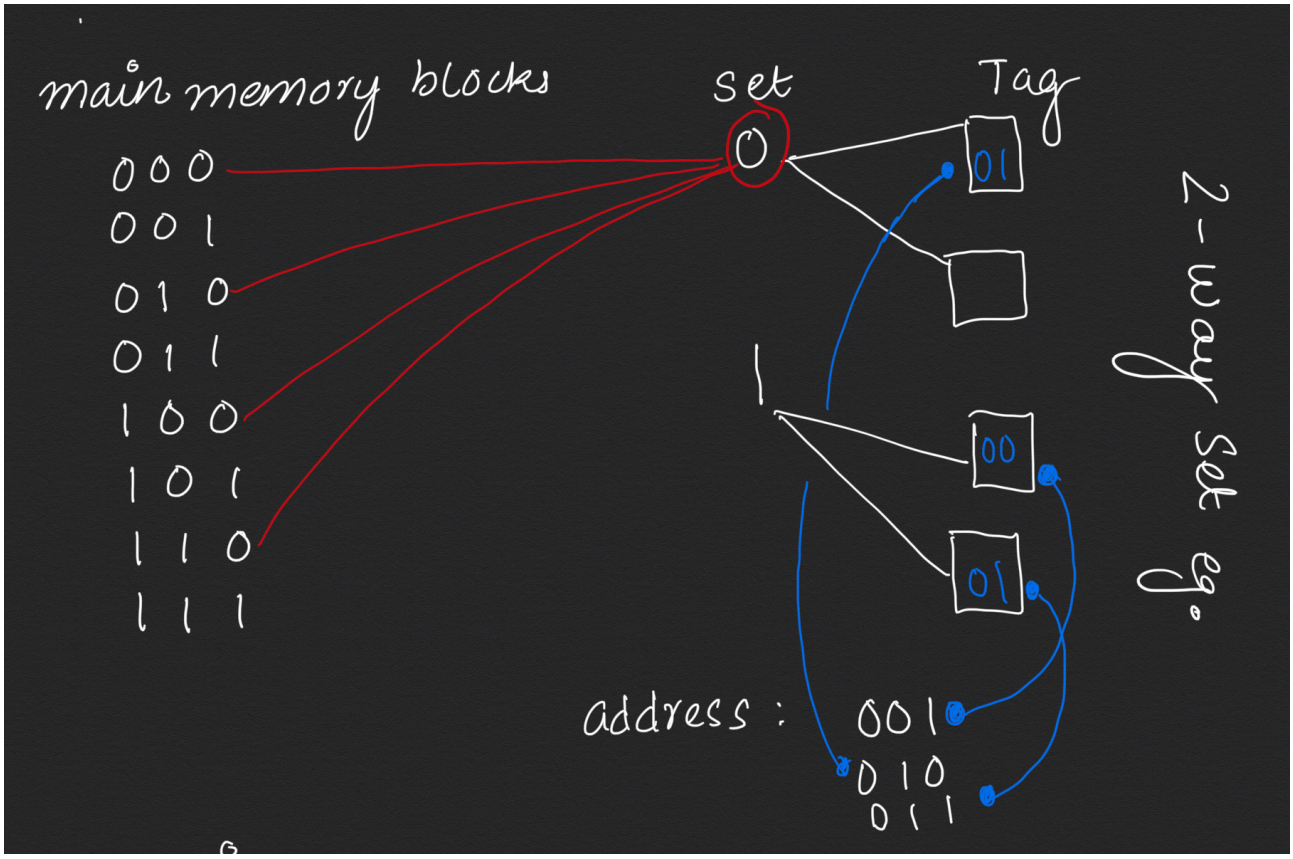
2) ASSOCIATIVE MAPPING:

Unlike Direct mapping, in case of associative mapping, a memory block is not associated to a single cache line. Instead, as requests are made by the CPU, the memory block corresponding to the address keeps getting added to the cache. When the cache is full, replacement of memory blocks in cache starts taking place. The replacement algorithm that I have used is First In First Out (FIFO). The memory block that is added first to the cache is the first one to be replaced. While searching the cache based on the request made by the CPU, a pointer is traversed in the cache memory to find the block in cache. If the block is found, it's a cache hit. If the block is not found in the cache, Its a cache miss. If it is a cache miss, the memory block requested by the CPU is added to the cache by following the same approach by which it was added earlier.



3) N-WAY SET ASSOCIATIVE MAPPING:

In this, the cache memory is divided into sets of N cache lines each. Which implies that the total number of sets would equal to Total Number of cache lines in cache/N. Based on the size inputs of main memory, cache and block, an array consisting of all the possible sets(along with number of elements(cache lines) in each set) is created for reference. Like in case of direct mapping, where each memory block is associated to a particular cache line, in this case, each memory block is associated to a particular set according to the LSB(s) of the the memory block address. Now, the cache lines in the sets are filled with memory block as and when requested to add to cache.A tag array with size same as the number of cache lines is created. When a memory block is to be added to cache, the address of the cache is broken down into Tag and Set. The Set is searched in the set array and the Tag is added to the corresponding index in the tag array. While searching, the address requested by the CPU is broken down into Tag, set, and Block offset. The set is searched in the set array. If found, a search is made in N elements of the tag array which correspond to that particular set. In this way, the search time is reduced, because instead of searching the complete array, just N elements of the tag array is searched which corresponds to a particular Set. This is a hybrid of Direct Mapping and Associative Mapping. Sets are mapped by Direct Mapping and Tags are mapped by Associative mapping.



BONUS PROBLEM FOR SECTION B:

Task - Build a 2 level cache. The size of the level 1 cache is S/2 and the size of level 2 cache is S. All the other parameters are to be decided accordingly. This will also be a standalone cache without the intervention of the main memory.

A level 2 cache is added to the Associative mapping approach. The number of cache lines in the level 2 cache is double the number of cache lines in Level 1 cache. When a memory block is to be added to the cache, its address is added to both Level 1 and Level 2 cache. When the level 1 cache becomes full, replacement of memory block is don by First in First Out algorithm. It is possible for a memory block to be present in Level 2 cache but not in Level 1 cache but the opposite is not possible. When both Level 1 and Level 2 cache memories are filled up, The same address is replaced from both the cache memories. While searching, Level 1 cache is searched first. If address if found in Level 1 cache, it's a cache hit. If it is not found in level 1 cache, the address is searched for in level 2 cache. If it is present in level 3 cache, it is added to level 1 cache and its a cache hit. If not found, it's a cache miss and the block is then added to both Level 1 and Level 2 cache memories.

SCREENSHOT OF OUTPUT SCREEN:

```
1. PRINT MAIN MEMORY
2.PRINT CACHE MEMORY
3.ADD TO CACHE MEMORY
4.SEARCH IN CACHE MEMORY
5.EXIT
3
ENTER ADDRESS OF BLOCK TO ADD TO CACHE0000
ADDING TO CACHE
 B0 ADDED TO CACHE L1 AND L2
['0000']
['0000']
1. PRINT MAIN MEMORY
2.PRINT CACHE MEMORY
3.ADD TO CACHE MEMORY
4.SEARCH IN CACHE MEMORY
5.EXIT
3
ENTER ADDRESS OF BLOCK TO ADD TO CACHE0001
ADDING TO CACHE
 B1 ADDED TO CACHE L1 AND L2
['0000', '0001']
['0000', '0001']
1. PRINT MAIN MEMORY
2.PRINT CACHE MEMORY
3.ADD TO CACHE MEMORY
4.SEARCH IN CACHE MEMORY
5.EXIT
4
BITS REQUIRES FOR PHYSICAL ADDRESS BITS: 7.0
ENTER PHYSICAL ADDRESS TO SEARCH IN CACHE MEMORY0001000
PHYSICAL ADDRESS: 0001000

CACHE ADDRESS: 0001

BLOCK OFFSET: 000

BLOCK INDEX: 0001

 B1 FOUND IN L1 CACHE
L1 CACHE HIT
1. PRINT MAIN MEMORY
2.PRINT CACHE MEMORY
3.ADD TO CACHE MEMORY
4.SEARCH IN CACHE MEMORY
5.EXIT
```

SUBMITTED BY:
AMAN PRIYADARSHI
2019294
SECTION B
CSD