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.

Solution

In this assignment ,I have implemented three different types of cache mapping.

The inputs to the program include S(size of cache),CL(number of cache lines),B(block size) and n(for n associative mapping).

Programming language:Java

ASSUMPTIONS

The following are the assumptions that we have taken while implementing the mappings:

1. *The number entered as input are integer binary numbers and all are powers of two .*
2. *The machine is a 16 bit machine.*

GETTING STARTED

Please follow these instructions to use the various types of mapping:

1. *Pick any number following constraints mentioned in the assumption.*
2. *Load the java program on to your compiler and hit the run button.*

3. *When the program will run you will know if there was a cache hit or miss.*

SUPPORT

If you have any issues regarding this project, feel free to contact me.
Manasvi Singh manasvi19369@iiitd.ac.in

UNDERSTANDING CACHE

Cache Memory is a special very high-speed memory. Cache memory is costlier than main memory but economical than CPU registers. Cache memory is an extremely fast memory type .It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.
The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations.

Cache Performance:

When the processor needs to read or write a location in main memory, it first checks for a corresponding entry in the cache.

- *If the processor finds that the memory location is in the cache:cache hit*
- *If the processor does not find the memory location in the cache:cache miss*

Cache Mapping:

There are three different types of mapping used for the purpose of cache memory which are as follows: Direct mapping, Associative mapping, and Set-Associative mapping. These are explained below.

1)Direct Mapping –
The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line. An address space is split into two parts: index field and a tag field.

2)Associative Mapping –

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.

3)Set-associative Mapping –

Instead of having exactly one line(like fully associative mapping) 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.In this case, the cache consists of a number of sets, each of which consists of a number of lines.

<u>FUNCTIONS AND THEIR EXPLANATION</u>

public static void main(String[] args)

The execution of the program begins with main().This takes the inputs and calls other specific functions.

DIRECT MAPPING:
public void write(Node[] tag_array,int idexing,String address,int val,int max_length,int index)

Since, here we have an assignment of each memory block to a specific line in the cache.
Int idexing gives an exact index of where the memory block has to be stored.
The address is divided into three parts:tag part,index part and offset.
This is very fast since the exact location is known.

public void read(Node[] tag_array,int ind,String address,int index)
This function is used to read from the cache array if an address is present (cache hit) .If not,cache miss.

FULLY ASSOCIATIVE MAPPING:
Any block can go into any line of cache.Therefore its the most flexible mapping form.
public void read(Node[] tag_array,String address,int index,LinkedList<String> q)
This function reads the value stored in the address if it's present,if it's not present then there is a cache miss.To read a value it has to search the whole cache till it finds the address or cache search is complete.If address is present then there's a cache hit.

public static int search(Node arr[], String x)
This function is used to find if any block has the address ‘x’.

public String write(Node[] tag_array,String address,int val,int max_length,int index,LinkedList<String> q)

If the block is already present then it just finds it and updates the word with the new value and it's a cache hit.
If not ,then there's a cache miss,if there’s vacant space in cache ,then a new block is added and val is stored.If the block is not present and there's no vacant space then the least recently used block is replaced with a new block.

<u>N way associative mapping:</u>

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.
public void read(Node[] tag_array,int ind,String tag,int address,int index,int noofblocks,LinkedList<String> q)
This function reads,it's given the set index and then it goes to the set index and checks the blocks in that set ,if any block's address is ‘tag’,then data value is read and there is a cache hit else if no block in set has ‘tag’ address ,there is cache miss.

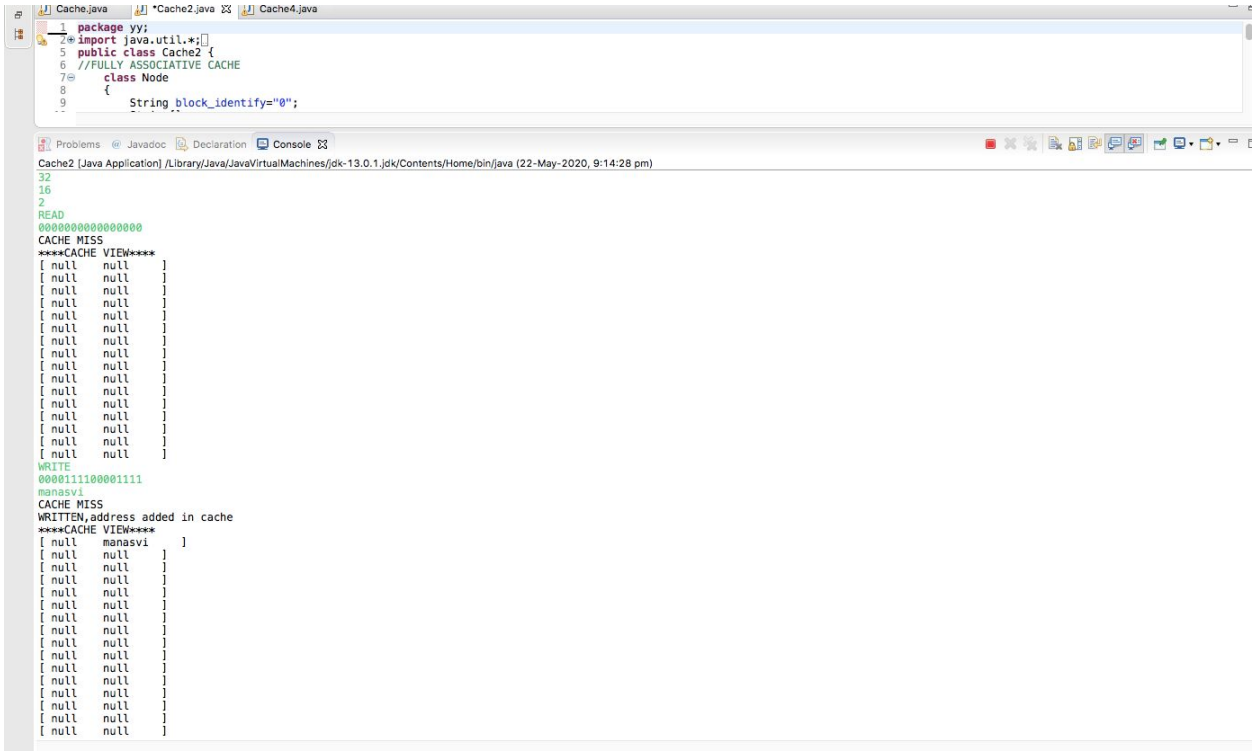
public void write(Node[] tag_array,int idexing,String tag,int address,int val,int max_length,int index,int noofblocks,LinkedList<String> q)

Set index is given ,if the address of any block within the set is ‘tag’,then the value of a particular word inside that block is updated and there’s a cache hit,if not there is a cache miss and if any entry in set is vacant then,the block is entered there.

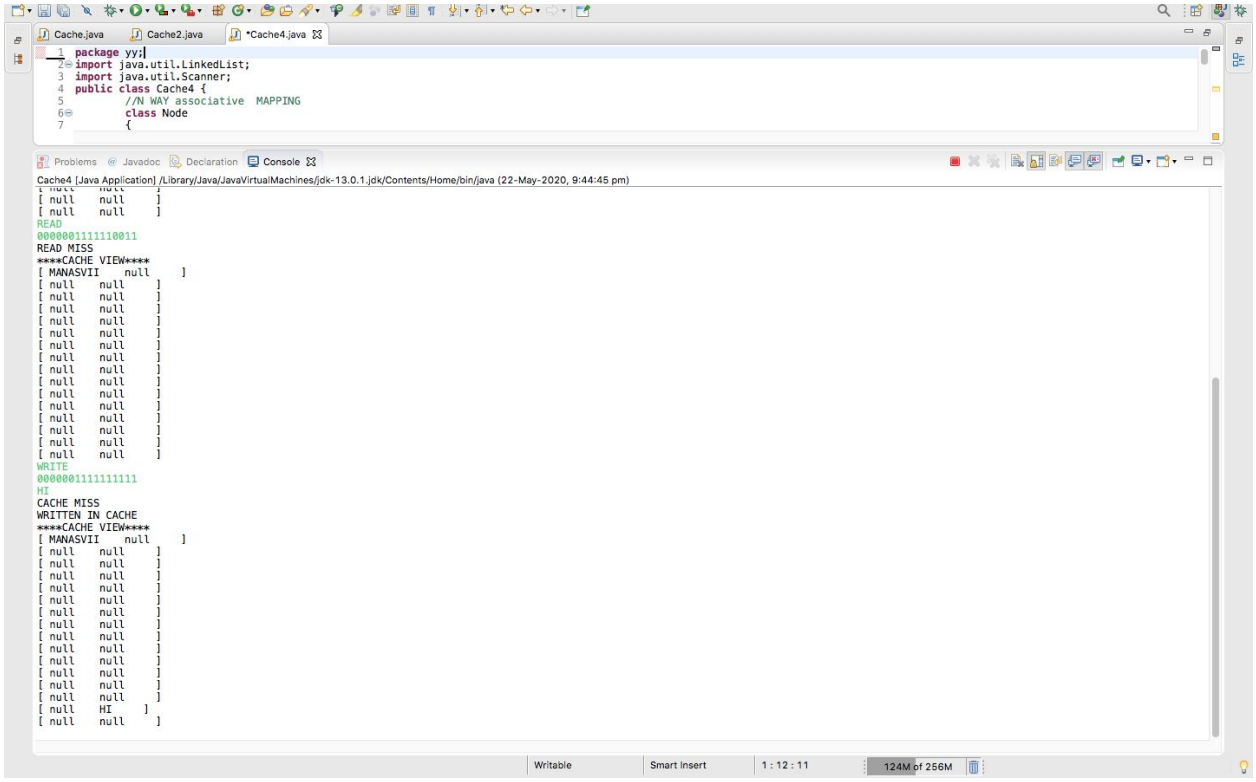
If there's no place vacant inside the set and the block is already not present,then the least recently used block in that block is replaced with the new block.

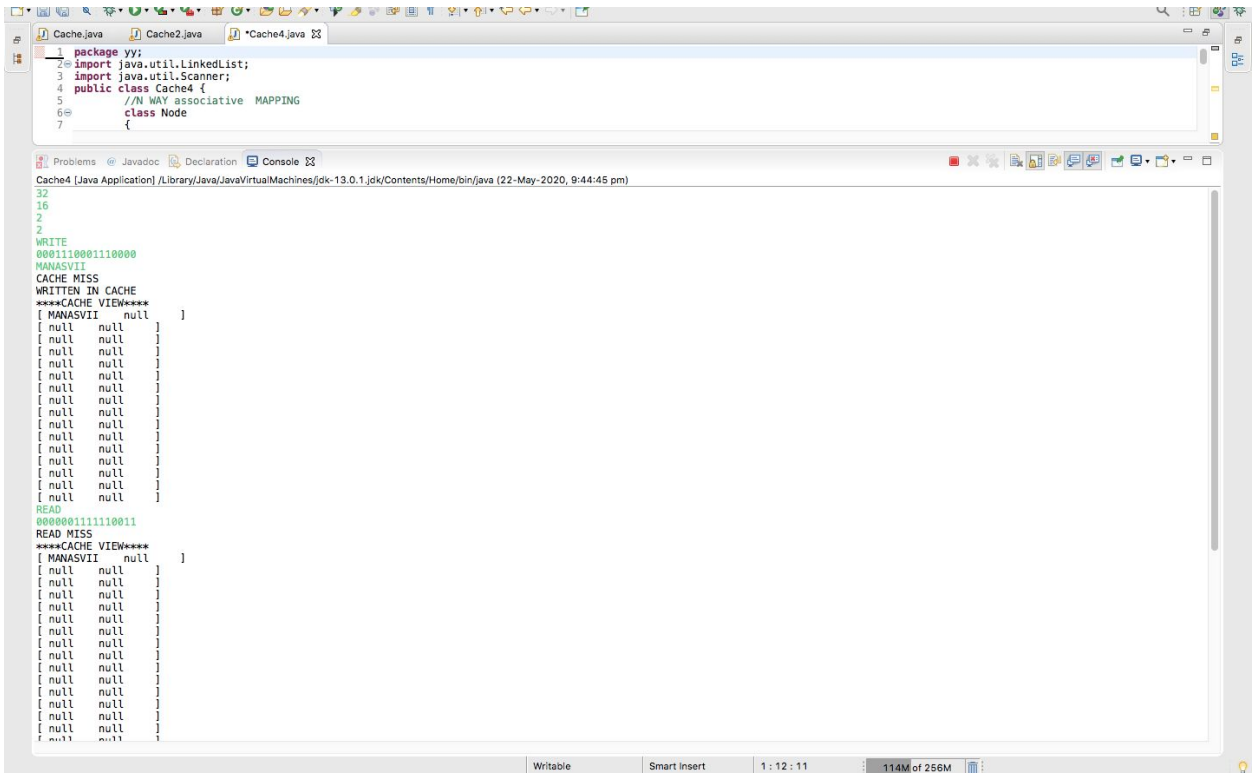
OUTPUTS

FULLY ASSOCIATIVE CACHE

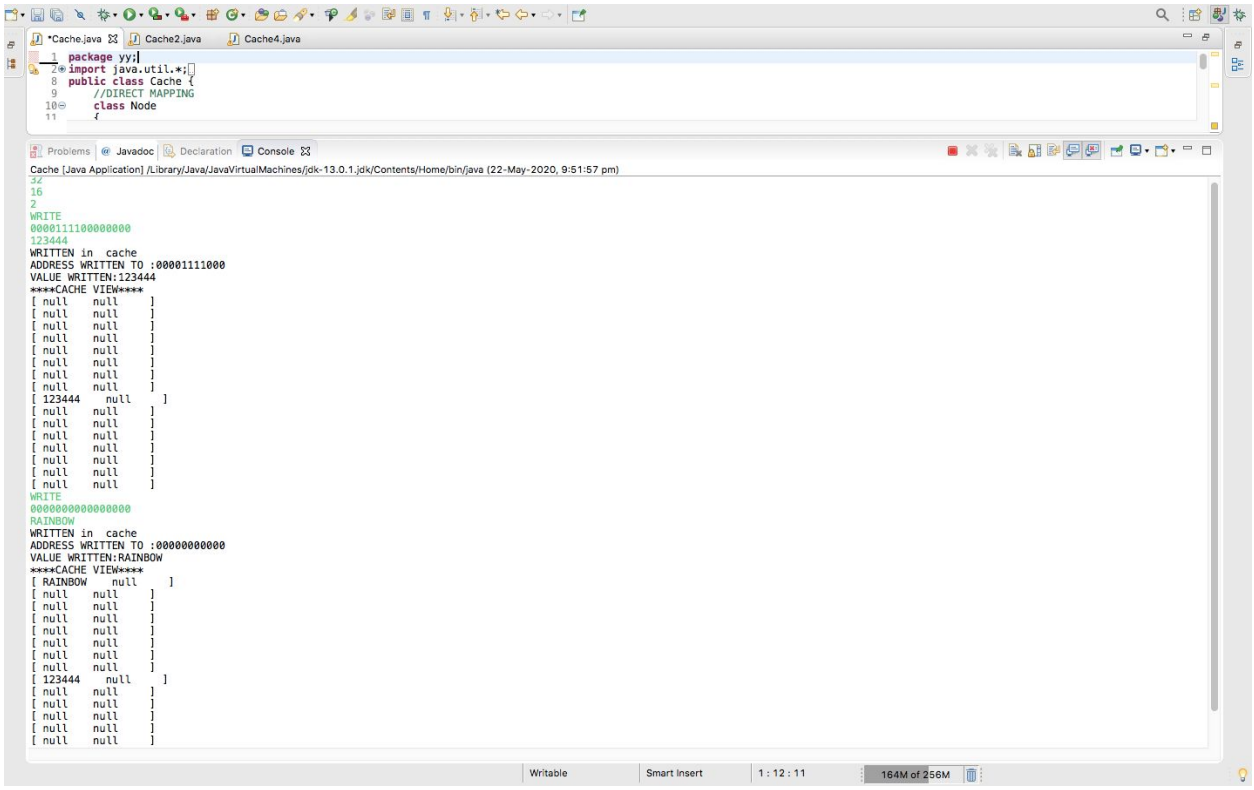


N way associative cache





DIRECT MAPPING


*Cache.java
Cache2.java
Cache4.java
1 package yy;
2 import java.util.*;
8 public class Cache {
9 //DIRECT MAPPING
10 class Node
11 {
Problems
Javadoc
Declaration
Console
Cache [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home/bin/java (22-May-2020, 9:51:57 pm)
32
16
2
WRITE
0000111100000000
123444
WRITTEN in cache
ADDRESS WRITTEN TO :00001111000
VALUE WRITTEN:123444
****CACHE VIEW****
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[123444 null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
WRITE
0000000000000000
RAINBOW
WRITTEN in cache
ADDRESS WRITTEN TO :00000000000
VALUE WRITTEN:RAINBOW
****CACHE VIEW****
[RAINBOW null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[null null]
[123444 null]
[null null]
[null null]
[null null]
[null null]
[null null]
Writable
Smart Insert
1 : 12 : 11
164M of 256M