**Batch: A1 Roll No.: 1911004**

**Experiment / assignment / tutorial No. 8**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE: Implementation of LRU Page Replacement Algorithm.** |

**AIM:** The LRU algorithm replaces the least recently used that is the last accessed memory block from user.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO 4-Learn and evaluate memory organization and cache structure

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

It follows a simple logic, while replacing it will replace that page which has least recently used out of all.

a) A hit is said to be occurred when a memory location requested is already in the cache.

b) When cache is not full, the number of blocks is added.

c) When cache is full, the block is replaced which is recently used

**Algorithm:**

1. Start
2. Get input as memory block to be added to cache
3. Consider an element of the array
4. If cache is not full, add element to the cache array
5. If cache is full, check if element is already present
6. If it is hit is incremented
7. If not, element is added to cache removing least recently used element
8. Repeat step 3 to 7 for remaining elements
9. Display the cache at very instance of step 8
10. Print hit ratio
11. End

**Code:**

import java.util.\*;

class LRU {

public static int min(int C[],int cacheBlockNo) {

int minimum = C[0];

int pos = 0;

for(int i=0;i<cacheBlockNo;i++)

if(minimum > C[i]) {

minimum = C[i];

pos = i;

}

return pos;

}

public static void main(String[] args) {

Scanner ob= new Scanner(System.in);

int n,recent = 0,cacheBlockNo;

System.out.print("Enter the number of main memo blocks ");

n = ob.nextInt();

int mainBlock[] = new int[n];

System.out.print("Enter the main memo block data ");

for(int i=0;i<n;i++)

mainBlock[i]=ob.nextInt();

System.out.print("\nEnter the number of cache blocks ");

cacheBlockNo = ob.nextInt();

int cacheBlock[] = new int[cacheBlockNo];

int C[] = new int[cacheBlockNo];

for(int i=0;i<cacheBlockNo;i++) {

cacheBlock[i] = 0;

C[i] = 0;//here 0 references an empty space in frame

}

System.out.println("Cache Content Iteration No");

for(int i=0;i<n;i++) {

int flag =0;

for(int j=0;j<cacheBlockNo;j++) {

if(cacheBlock[j] == mainBlock[i]) {

flag=1;

C[j] = recent++;

break;

}

}

if(flag == 0) {

for(int j=0;j<cacheBlockNo;j++){

if(cacheBlock[j] == 0){

cacheBlock[j] = mainBlock[i];

C[j] = recent++;

flag=1;

break;

}

}

}

if(flag == 0 ){

int PTR = min(C,cacheBlockNo);

cacheBlock[PTR] = mainBlock[i];

C[PTR] = recent++;

}

System.out.println("");

for(int j=0;j<cacheBlockNo;j++) {

System.out.print(cacheBlock[j]+" ");

}

System.out.print("\t \t \t "+(i+1));

if(i==0 )

System.out.print(" HERE 1ST ELEMENT IS FROM MAIN MEMO REST IS GARBAGE")

else if(i==1)

System.out.print(" HERE 1ST 2 ELEMENT ARE FROM MAIN MEMO REST IS GARBAGE")

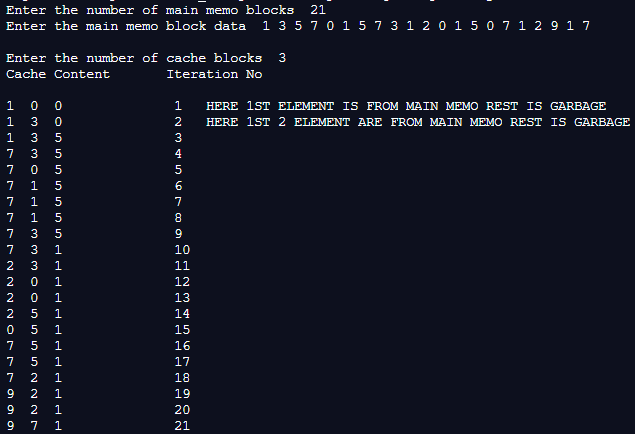
}

System.out.println("\n");

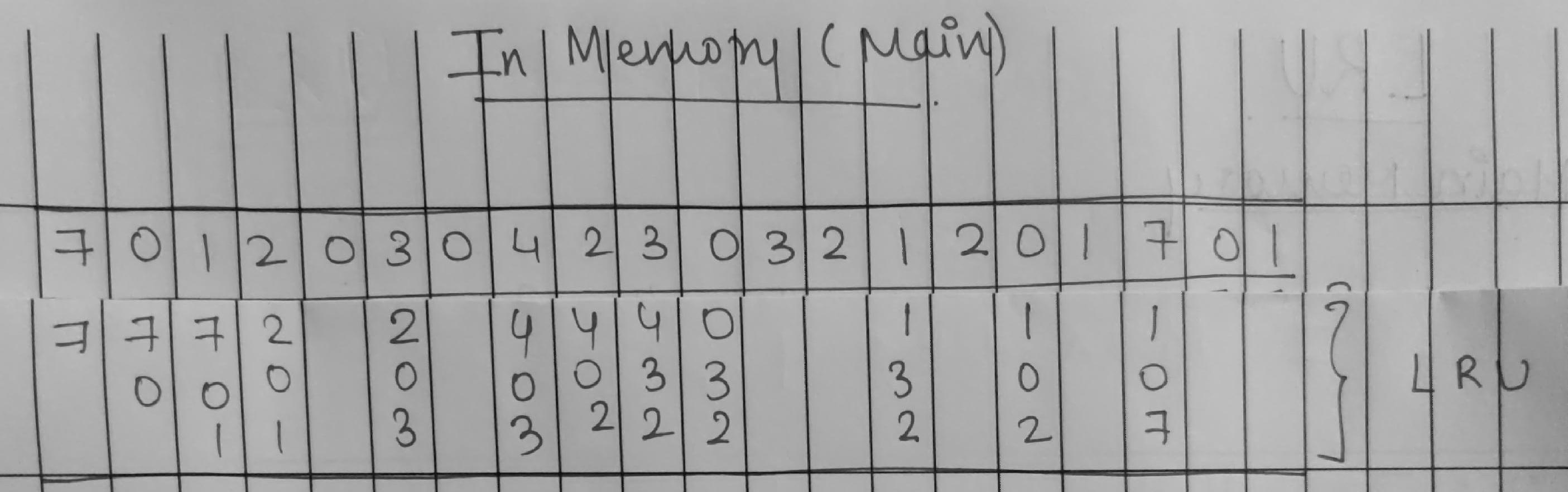
}

}

**Output :**



**Example:**



**Post Lab Descriptive Questions**

1. **Define hit rate and miss ratio?**

A hit ratio is a ratio calculation of cache hits is to how many total content requests were received.

A miss ratio is a ratio of the cache misses compared to the total number of content requests that were received.

**2. What is the need for virtual memory**?

Virtual Memory is section of the hard disk is used to store items of RAM which are not being currently used .

When a computer is running the operating system and several other programs at the same instance of time , the physical memory often might become full.

Instead of closing some programs the operating system will ‘pretend’ or use virtual memory concept to store some of the data. Thus showing that it has infinite memory. Used to allow more programs / data to be loaded ,stored when the RAM is insufficient**.**

**Conclusion : We understood the LRU Page Replacement Algorithm & implemented it in java**

**Date: 13-10-2020 Signature of faculty in-charge**

**Batch: A1 Roll No.: 1911004**

**Experiment / assignment / tutorial No. 9**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE :**Implementation ofFIFO Page Replacement Algorithm |

**AIM:** The FIFO algorithm uses the principle that the block in the set which has been in for the longest time will be replaced

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO 4-Learn and evaluate memory organization and cache structure

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, TataMcGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

**3**. Dr. M. Usha, T. S. Srikanth, “Computer System Architecture and Organization”, First Edition, Wiley-India.

**---------------------------------------------------------------------------------------------------------**

**Pre Lab/ Prior Concepts:**

T he FIFO algorithm uses the principle that the block in the set which has been in the block for the longest time is replaced. FIFO is easily implemented as a round robin or criteria buffer technique. The data structure used for implementation is a queue. Assume that the number of cache pages is three. Let the request to this cache is shown alongside.

**Algorithm:**

1. A hit is said to be occurred when a memory location requested is already in the cache.

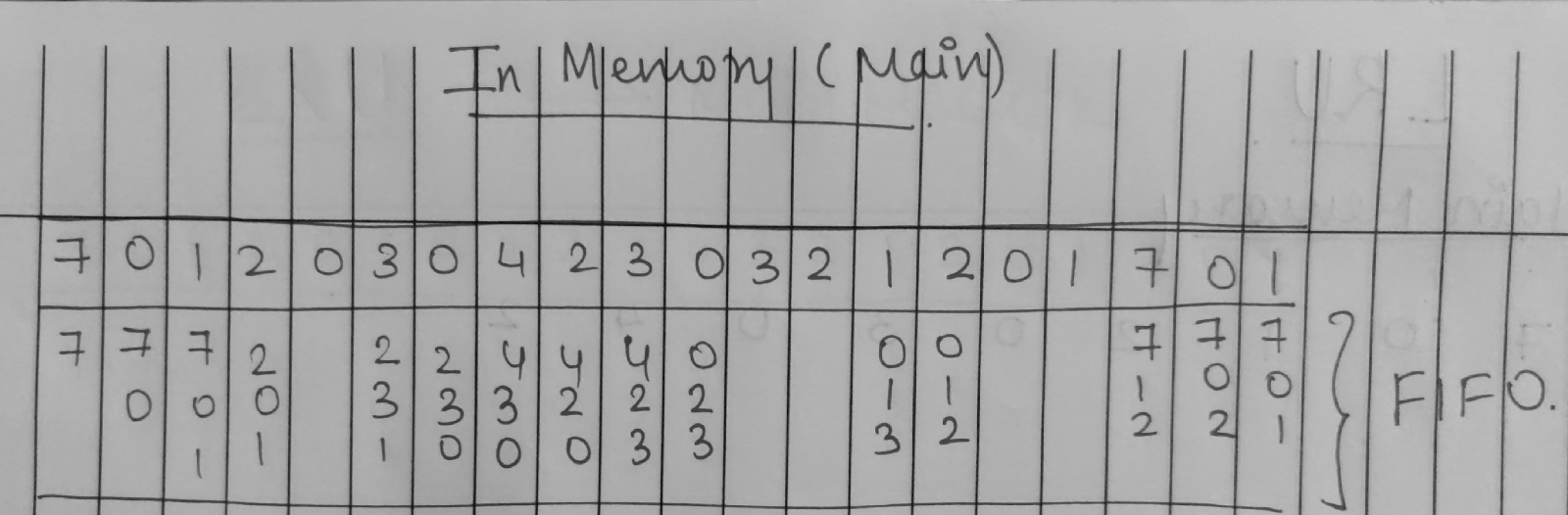
2. When cache is not full, the number of blocks is added.

3. When cache is full, the block is replaced which was added first

**Design Steps:**

1. Start
2. Get input as memory block to be added to cache
3. Consider an element of the array
4. If cache is not full, add element to the cache array
5. If cache is full, check if element is already present
6. If it is hit is incremented
7. If not, element is added to cache removing first element (which is in first).
8. Repeat step 3 to 7 for remaining elements
9. Display the cache at very instance of step 8
10. Print hit ratio
11. End.

**Example:**



**Code:**

**import java.util.\*;**

**class FIFO {**

**public static void main(String args[]) {**

**int fifo[]; //fifo cache array**

**int n,hit = 0,j= 0,i= 0,k= 0;**

**boolean check;**

**Scanner ob = new Scanner(System.in);**

**System.out.print("Enter the number of main memo block ");**

**n = ob.nextInt();**

**int mainMemo[] = new int[n];**

**System.out.print("Enter the elements ");**

**for (i = 0; i < n; i++)**

**mainMemo[i] = ob.nextInt();**

**System.out.print("\nEnter no of cache block ");**

**fifo = new int[ob.nextInt()];**

**System.out.println("\nCache Content Iteration No");**

**for (i = 0; i < n; i++) {**

**check = false;**

**for (k = 0; k < 3; k++)**

**if (fifo[k] == mainMemo[i]) {**

**check = true;**

**hit ++;}**

**if (check == false) {**

**fifo[j] = mainMemo[i];**

**j++;**

**if (j >= 3)**

**j = 0; }**

**for(k=0;k<3;k++)**

**System.out.print(fifo[k] + " ");**

**System.out.print("\t \t \t "+(i+1)); //printing the cache elements**

**if(i==0 )**

**System.out.print("\tHERE 1ST ELEMENT IS FROM MAIN MEMO REST IS GARBAGE\n");**

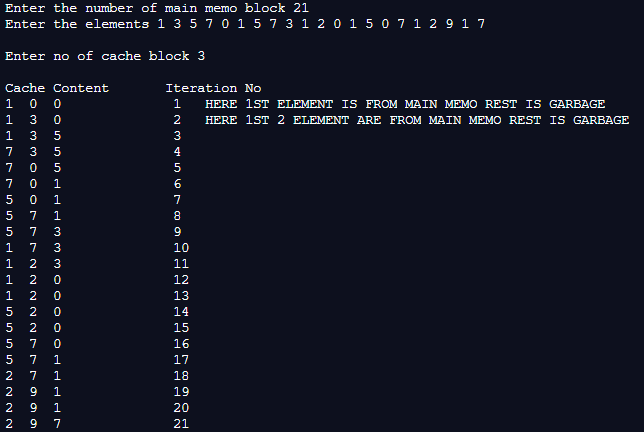
**else if(i==1)**

**System.out.print("\tHERE 1ST 2 ELEMENT ARE FROM MAIN MEMO REST IS GARBAGE\n");**

**}**

**}**

**}**

**Output**

**Post Lab Descriptive Questions**

**1. What is meant by memory interleaving?**

It is a technique for compensating the relatively slow speed of DRAM .In this technique, the main memory is divided into memory banks which can be accessed individually without any dependency on the other.

Thus in short memory interleaving is process of dividing memory into a number of modules such that Successive words in the address space are placed in the Different module.

We use this when cache is missing the needed data thus it has to retrieve it from main memory in such cases we use Memory Interleaving .

**2. Explain Paging Concept?**

**Paging** is a memory management mechanism that allows OS to store & retrieve data/instructions/processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames/pages. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation.

Paging is technique used for faster & easy access to data

**Conclusion : We understood the FIFO Page Replacement Algorithm & implemented it in java**

**Date: 13-10-2020 Signature of faculty in-charge**