K. J. Somaiya College of Engineering, Mumbai-77

Batch: B1 **Roll No.: 1711072**

Experiment / assignment / tutorial No. 6

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.

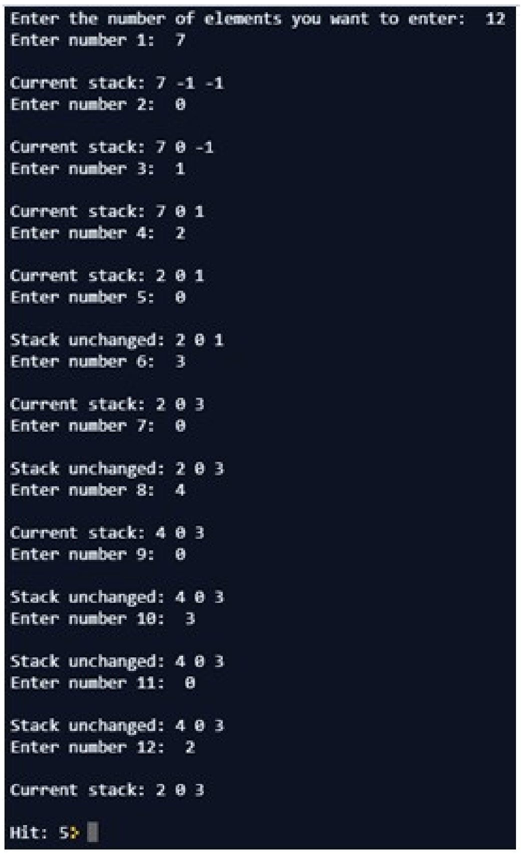
c) When cache is full, the block is replaced which is recently used

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

Example:



```
Enter the number of elements you want to enter:  12
Enter number 1:  7

Current stack: 7 -1 -1
Enter number 2:  0

Current stack: 7 0 -1
Enter number 3:  1

Current stack: 7 0 1
Enter number 4:  2

Current stack: 2 0 1
Enter number 5:  0

Stack unchanged: 2 0 1
Enter number 6:  3

Current stack: 2 0 3
Enter number 7:  0

Stack unchanged: 2 0 3
Enter number 8:  4

Current stack: 4 0 3
Enter number 9:  0

Stack unchanged: 4 0 3
Enter number 10:  3

Stack unchanged: 4 0 3
Enter number 11:  0

Stack unchanged: 4 0 3
Enter number 12:  2

Current stack: 2 0 3

Hit: 5>
```

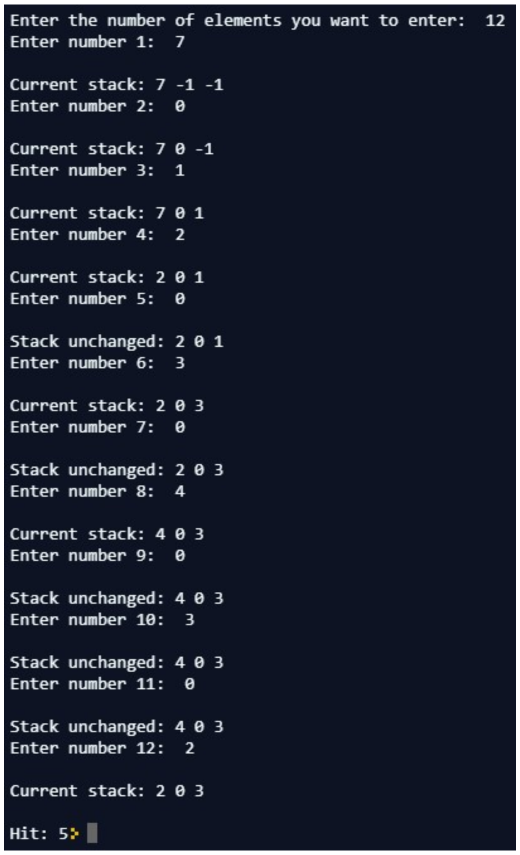
Implementation Details (in Java):

```
import java.util.*;

class Main {
  public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    int len=3;
    int elem, hit=0;
    int arr[]=new int[]{-1,-1,-1};
    int priority[]=new int[]{3,3,3};
    System.out.print("Enter the number of elements you want to enter: 
");
    int n=sc.nextInt();
    for(int i=0;i<n;i++){
      System.out.print("Enter number "+(i+1)+": ");
       elem=sc.nextInt();
      int index=search(elem, arr);
      if(index!=-1){
        System.out.print("\nStack unchanged: ");
        for(int j=0;j<len;j++){
          if(priority[j]!=3)
            priority[j]+=1;
          priority[search(elem,arr)]=1;
        }
        hit++;
        print(arr);
      }
      else{
        arr[search(3,priority)]=elem;
        System.out.print("\nCurrent stack: ");
        for(int j=0;j<len;j++){
          if(priority[j]!=3){
            priority[j]+=1;
          }
        }
        priority[search(elem,arr)]=1;
        print(arr);
      }
      if(i==n-1)
        System.out.println("Hit: "+hit);
    }
    sc.close();
  }
  static int search(int elem, int[] arr){
    for(int i=0;i<arr.length;i++){
      if(arr[i]==elem){
        return i;
      }
    }
    return -1;
  }
  static void print(int[] arr){
    for(int i=0;i<arr.length;i++){
    System.out.print(arr[i]+" ");
    System.out.println();
    }
  }
}
```

For verification, my code is available on:
https://repl.it/@ARGHYADEEPDAS/LRUPageReplacementAlgorithm

Output Screen:



```
Enter the number of elements you want to enter:  12
Enter number 1:  7

Current stack: 7 -1 -1
Enter number 2:  0

Current stack: 7 0 -1
Enter number 3:  1

Current stack: 7 0 1
Enter number 4:  2

Current stack: 2 0 1
Enter number 5:  0

Stack unchanged: 2 0 1
Enter number 6:  3

Current stack: 2 0 3
Enter number 7:  0

Stack unchanged: 2 0 3
Enter number 8:  4

Current stack: 4 0 3
Enter number 9:  0

Stack unchanged: 4 0 3
Enter number 10:  3

Stack unchanged: 4 0 3
Enter number 11:  0

Stack unchanged: 4 0 3
Enter number 12:  2

Current stack: 2 0 3

Hit: 5> 
```

Post Lab Descriptive Questions (Add questions from examination point view)

1. Define hit ratio and miss ratio?

2. What is the need for virtual memory?

1. Hit rate is the fraction of accesses that were a hit.

   Hit rate=Hits/(Hits+Misses)

   Miss ratio is the fraction of accesses that were a miss.

   Miss ratio=Misses/(Hits+Misses)

The (hit/miss) latency (AKA access time) is the time it takes to fetch the data in case of a hit/miss. If the access was a hit - this time is rather short because the data is already in the cache. But if it was a miss - that time is much linger as the (slow) L3 memory needs to be accessed. The latency depends on the specification of your machine: the speed of the cache, the speed of the slow memory, etc.

Remember,

Hit ratio+Miss ratio=1

2. If the computer lacks the random access memory (RAM) needed to run a program or operation, Windows uses virtual memory to compensate. Virtual memory combines the computer's RAM with temporary space on your hard disk. When RAM runs low, virtual memory moves data from RAM to a space called a paging file. So basically, there's a separate paging file loaded on your hard drive/solid state drive (ROM) when you run out of RAM.

Conclusion:

The program ran successfully as we were able to simulate the LRU page replacement algorithm in Java using a stack of size 3.

Date: 26/09/2018 **Signature of faculty in-charge**