CSE2008 (Operating Systems) Lab-8

KHAN MOHD OWAIS RAZA
20BCD7138

Q.1 Write a program for following page replacement algorithms and calculate page fault ration for each

a. FIFO
b. Optimal
c. Least recently used

Program must be menu driven where the user can select any replacement algorithm.

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138*/
/* CSE2008 (OPERATING SYSTEMS) LAB PRACTICAL-8 */
/* Write the program for following page replacement
   algorithms & calculate page fault ratio for each
   a) FIFO
   b) Optimal
   c) Least recently used
*/
package owaisraza.CSE2008_Lab8;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Iterator;
import java.io.*;
import java.util.*;
import java.util.Scanner;
public class MyClass{
static int Page_Fault(int PAGE[], int X, int CAPACITY){
HashSet<Integer> Y = new HashSet<>(CAPACITY);
Queue<Integer> INDEX = new LinkedList<>() ;
int Page_Fault = 0;
for (int A=0; A<X; A++){
if (Y.size() < CAPACITY){
if (!Y.contains(PAGE[A])){
Y.add(PAGE[A]);
Page_Fault++;
INDEX.add(PAGE[A]);
}}
else{
if (!Y.contains(PAGE[A])){
int val = INDEX.peek();
INDEX.poll();
Y.remove(val);
Y.add(PAGE[A]);
INDEX.add(PAGE[A]);
Page_Fault++;
}}
```

```
}
return Page_Fault;
}
static boolean search(int KEY, int[] F){
for (int A = 0; A < F.length; A++)
if (F[A] == KEY)
return true;
return false;
}
static int predict(int PAGE[], int[] F, int P, int INDEX){
int R = -1, FAR = INDEX;
for (int A = 0; A < F.length; A++) {
int B;
for (B = INDEX; B < P; B++) {
if (F[A] == PAGE[B]) {
if (B > FAR) {
FAR = B;
R = A;
}
break;
}}
if (B == P)
return A;
}
return (R == -1) ? 0 : R;
}
static void OPTIMAL_PAGE(int PAGE[], int P, int F2){
int[] F1 = new int[F2];
int HIT = 0;
int INDEX = 0;
for (int A = 0; A < P; A++) {
if (search(PAGE[A], F1)) {
HIT++;
continue;
}
if (INDEX < F2)
F1[INDEX++] = PAGE[A];
else {
int j = predict(PAGE, F1, P, A+1);
F1[j] = PAGE[A];
}
}
System.out.println("NUMBER OF HITS = " + HIT);
System.out.println("NUMBER OF MISSES = " + (P - HIT));
}
static int PF(int PAGE[], int X, int CAPACITY){
HashSet<Integer> Y = new HashSet<>(CAPACITY);
HashMap<Integer, Integer> INDEX = new HashMap<>();
int PF = 0;
for (int A=0; A<X; A++){
if (Y.size() < CAPACITY){
if (!Y.contains(PAGE[A])){
Y.add(PAGE[A]);
PF++;
```

```
}
INDEX.put(PAGE[A], A);
}
else{
if (!Y.contains(PAGE[A])){
int LEAST_RECENTLY_USED = Integer.MAX_VALUE, VALUE=Integer.MIN_VALUE;
Iterator<Integer> ITERATE = Y.iterator();
while (ITERATE.hasNext()) {
int temp = ITERATE.next();
if (INDEX.get(temp) < LEAST_RECENTLY_USED){
LEAST_RECENTLY_USED = INDEX.get(temp);
VALUE = temp;
}}
Y.remove(VALUE);
INDEX.remove(VALUE);
Y.add(PAGE[A]);
PF++;
}
INDEX.put(PAGE[A], A);
}}
return PF;
}
public static void main(String args[]){
int[] PAGE = new int[8];
Scanner sc = new Scanner(System.in);
for(int A =0; A<8; A++){
System.out.println("ENTER ELEMENT OF INDEX-" +A +":");
PAGE[A]=sc.nextInt();
}
int CAPACITY = 4;
System.out.println("(1) FIFO: ");
System.out.println(Page_Fault(PAGE, PAGE.length, CAPACITY));
System.out.println();
int P = PAGE.length;
int F2 = 4;
System.out.println("(2) OPTIMAL: ");
OPTIMAL_PAGE(PAGE, P, F2);
System.out.println();
System.out.println("(3) LEAST RECENTLY USED: ");
System.out.println(PF(PAGE, PAGE.length, CAPACITY));
}}
```

Output :-

```
ENTER ELEMENT OF INDEX-2:
23
ENTER ELEMENT OF INDEX-3:
6
ENTER ELEMENT OF INDEX-4:
8
ENTER ELEMENT OF INDEX-5:
9
ENTER ELEMENT OF INDEX-6:
4
ENTER ELEMENT OF INDEX-7:
8
(1) FIFO:
7

(2) OPTIMAL:
NUMBER OF HITS = 1
NUMBER OF MISSES = 7

(3) LEAST RECENTLY USED:
7
```