**Experiment 5**

**AIM:** To implement various page replacement techniques.

a) FIFO

b) LRU

c) OPTIMAL

**THEORY:** Page replacement algorithms are an important part of virtual memory management and it helps the operating system to decide which memory page can be moved out making space for the currently needed page. However, the ultimate objective of all page replacement algorithms is to reduce the number of page faults.

FIFO - This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

LRU - In this algorithm page will be replaced which is least recently used

OPTIMAL - In this algorithm, pages are replaced which would not be used for the longest duration of time in the future. This algorithm will give us less page fault when compared to other page replacement algorithms.

**A) FIRST IN FIRST OUT SOURCE CODE :**

#include<stdio.h>

#include<conio.h>

int fr[3];

void main() {

void display();

int i,j,page[12]={2,3,2,1,5,2,4,5,3,2,5,2};

int flag1=0,flag2=0,pf=0,frsize=3,top=0;

clrscr();

for(i=0;i<3;i++) {

fr[i]= -1;

}

for(j=0;j<12;j++) {

flag1=0;

flag2=0;

for(i=0;i<12;i++) {

if(fr[i]==page[j]) {

flag1=1;

flag2=1;

break;

}

}

if(flag1==0) {

for(i=0;i=frsize)

top=0;

}

display();

}

printf("Number of page faults : %d ",pf+frsize);

getch();

}

void display() {

int i;

printf("\n");

for(i=0;i<3;i++) printf("%d\t",fr[i]);

}

**OUTPUT:**

2 1 1

2 3 1

2 3 1

2 3 1

5 3 1

5 2 1

5 2 4

5 2 4

3 2 4

3 2 4

3 5 4

3 5 2

Number of page faults: 9

**B) LEAST RECENTLY USED SOURCE CODE :**

#include<stdio.h>

#include<conio.h>

int fr[3];

void main() {

void display();

int p[12]={2,3,2,1,5,2,4,5,3,2,5,2},i,j,fs[3];

int index,k,l,flag1=0,flag2=0,pf=0,frsize=3;

clrscr();

for(i=0;i<3;i++) {

fr[i]=-1;

}

for(j=0;j<12;j++) {

flag1=0,flag2=0;

for(i=0;i<3;i++) {

if(fr[i]==p[j]) {

flag1=1;

flag2=1;

break;

}

}

if(flag1==0) {

for(i=0;i<3;i++) {

if(fr[i]== -1) {

fr[i]=p[j];

flag2=1;

break;

}

}

}

if(flag2==0) {

for(i=0;i<3;i++) fs[i]=0;

for(k=j -1,l=1;l<=frsize -1;l++,k-- ) {

for(i=0;i<3;i++) {

if(fr[i]==p[k]) fs[i]=1;

}

}

for(i=0;i<3;i++) {

if(fs[i]==0) index=i;

}

fr[index]=p[j];

pf++;

}

display();

}

printf(" \n no of page faults :%d",pf+frsize);

getch();

}

void display() {

int i;

printf(" \ n");

for(i=0;i<3;i++) printf(" \ t%d",fr[i]);

}

**OUTPUT:**

2 1 1

2 3 1

2 3 1

2 3 1

2 5 1

2 5 1

2 5 4

2 5 4

3 5 4

3 5 2

3 5 2

3 5 2

No of page faults: 7

**C) OPTIMAL SOURCE CODE:**

#include<stdio.h>

#include<conio.h>

int fr[3], n, m;

void display();

void main()

{

int i,j,page[20],fs[10];

int max,found=0,lg[3],index,k,l,flag1=0,flag2=0,pf=0;

float pr;

clrscr();

printf("Enter length of the reference string: ");

scanf("%d",&n);

printf("Enter the reference string: ");

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

scanf("%d",&page[i]);

printf("Enter no of frames: ");

scanf("%d",&m);

for(i=0;i<m;i++)

fr[i]=-1;

pf=m;

for(j=0;j<n;j++) {

flag1=0;

flag2=0;

for(i=0;i<m;i++) {

if(fr[i]==page[j]) {

flag1=1;

flag2=1;

break;

}

}

if(flag1==0) {

for(i=0;i<m;i++) {

if(fr[i] == -1){

fr[i]=page[j];

flag2=1;

break;

}

}

}

if(flag2==0) {

for(i=0;i<m;i++)

lg[i]=0;

for(i=0;i<m;i++) {

for(k=j+1;k<=n;k++) {

if(fr[i]==page[k]) {

lg[i]=k-j;

break;

}

}

}

found=0;

for(i=0;i<m;i++) {

if(lg[i]==0) {

index=i;

found = 1;

break;

}

}

if(found==0) {

max=lg[0];

index=0;

for(i=0;i<m;i++) {

if(max<lg[i]) {

max=lg[i];

index=i;

}

}

}

fr[index]=page[j];

pf++;

}

display();

}

printf("Number of page faults : %d\n", pf);

pr=(float)pf/n\*100;

printf("Page fault rate = %f \n", pr); getch();

}

void display()

{

int i;

for(i=0;i<m;i++)

printf("%d\t",fr[i]);

printf("\n");

}

**OUTPUT:**

Enter length of the reference string: 12

Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5

Enter no of frames: 3

1 -1 -1

1 2 -1

1 2 3

1 2 4

1 2 4

1 2 4

1 2 5

1 2 5

1 2 5

3 2 5

4 2 5

4 2 5

Number of page faults : 7

Page fault rate = 58.333332