8. PAGE REPLACEMENT TECHNIQUES – I

 1. write a C program to implement the FIFO page replacement algorithm

. 2. write a C program to implement the LRU page replacement algorithm

AIM: To implement FIFO page replacement technique.

a) FIFO b) LRU

DESCRIPTION:

Page replacement algorithms are an important part of virtual memory management and it helps the OS 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.

ALGORITHM:

1. Start the process

2. Read number of pages n

3. Read number of pages no

4. Read page numbers into an array a[i]

5. Initialize avail[i]=0 .to check page hit

6. Replace the page with circular queue, while re-placing check page availability in the frame

Place avail[i]=1 if page is placed in the frame Count page faults

7. Print the results.

8. Stop the process.

**1) 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;

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;i++)

{

if(fr[i]==-1)

{

fr[i]=page[j]; flag2=1; break;

}

}

}

if(flag2==0)

{

fr[top]=page[j];

top++;

pf++;

if(top>=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

2) LEAST RECENTLY USED

AIM: To implement LRU page replacement technique.

ALGORITHM:

1. Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8. Display the values

9. Stop the process

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