**Practical : 13**

**Aim: Implementation Of LRU (Least Recently Used) Page Replacement Algorithm.**

**Program:**

#include<stdio.h>

int findLRU(int time[], int n);

int main()

{

int frame\_size, no\_of\_pages, counter = 0, flag1,f, flag2, i, j, m, pos, page\_faults = 0;

int frames[10], pages[30], time[10], pri[20][20];

char x[20];

float fault\_ratio,hit\_ratio;

printf("Enter Total number of pages: ");

scanf("%d", &no\_of\_pages);

printf("Enter value of page number: ");

printf("\n");

for(i = 0; i < no\_of\_pages; i++)

{

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

}

printf("\n");

printf("Enter number of frames: ");

scanf("%d", &frame\_size);

for(i=0; i< no\_of\_pages; i++)

{

x[i]='X';

}

for(i = 0; i < frame\_size; ++i)

{

frames[i] = 0;

}

for(i = 0; i < no\_of\_pages; ++i)

{

flag1 = flag2 = 0;

for(j = 0; j < frame\_size; ++j)

{

if(frames[j] == pages[i])

{

counter++;

time[j] = counter;

flag1 = flag2 = 1;

page\_faults--;

x[i]='\*';

page\_faults++;

break;

}

}

if(flag1 == 0)

{

for(j = 0; j < frame\_size; ++j)

{

if(frames[j] == 0)

{

counter++;

page\_faults++;

frames[j] = pages[i];

time[j] = counter;

flag2 = 1;

break;

}

}

}

if(flag2 == 0)

{

pos = findLRU(time, frame\_size);

counter++;

page\_faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

for(j = 0; j < frame\_size; j++)

{

pri[j][i]=frames[j];

}

}

printf("\n");

for(i = 0 ; i <no\_of\_pages\*5+2\*frame\_size+1; i ++)

{

printf("-");

}

printf("\n| |");

for(i=1;i<=(no\_of\_pages\*4)/2;i++)

{

printf(" ");

}

printf("Pages");

for(i=(no\_of\_pages\*4)/2;i<=(no\_of\_pages\*4)+13;i++)

{

printf(" ");

}

printf("|\n");

printf("|Frames |");

for(i = 0 ; i < no\_of\_pages\*5+2\*frame\_size - 8; i ++)

{

printf("-");

}

printf("\n|\t|");

for(i=0 ; i < no\_of\_pages ; i++)

{

printf(" %2d |",pages[i]);

}

printf("\n");

for(i = 0 ; i < no\_of\_pages\*5+2\*frame\_size+1 ; i ++)

{

printf("-");

}

printf("\n");

for(i=0;i<frame\_size;i++)

{

printf("| %3d",i);

printf("\t|");

for(f=0;f<no\_of\_pages;f++)

{

if(pri[i][f]==0)

{

printf(" - |");

}

else

{

printf(" %2d |",pri[i][f]);

}

}

printf("\n");

}

for(i = 0 ; i < no\_of\_pages\*5+2\*frame\_size+1; i ++)

{

printf("-");

}

printf("\n| |");

for(i = 0; i< no\_of\_pages; i++)

{

if(x[i]=='X')

{

printf("\033[0;31m");

printf(" %2c ",x[i]);

printf("\033[0m");

printf("|");

}

else

{

printf("\033[0;32m");

printf(" %2c ",x[i]);

printf("\033[0m");

printf("|");

}

}

printf("\n");

for(i = 0 ; i < no\_of\_pages\*5+2\*frame\_size+1; i ++)

{

printf("-");

}

printf("\n\n Total Page Faults = Total No of pages - Total Pages hits \n");

printf(" = %d - %d \n",no\_of\_pages,(no\_of\_pages-page\_faults));

printf(" = %d \n",page\_faults);

printf("\n Total Page Hits = Total No of pages - Total Pages Miss \n ");

printf(" = %d - %d \n",no\_of\_pages,page\_faults);

printf(" = %d \n",(no\_of\_pages-page\_faults));

printf("\nTotal Page Fault ratio = Total Page faults / Total pages \n");

printf(" = %d / %d \n",page\_faults,no\_of\_pages);

printf(" = %5.2f \n",((float)page\_faults/no\_of\_pages));

printf("\nTotal Page Hit ratio = Total Page hits / Total pages \n");

printf(" = %d / %d \n",(no\_of\_pages-page\_faults),no\_of\_pages);

printf(" = %5.2f \n",((float)no\_of\_pages-page\_faults)/no\_of\_pages);

return 0;

}

int findLRU(int time[], int n)

{

int i, minimum = time[0], pos = 0;

for(i = 1; i < n; ++i)

{

if(time[i] < minimum)

{

minimum = time[i];

pos = i;

}

}

return pos;

}

**Output:**

