A MINI PROJECT REPORT

ON

LRU Page Replacement Algorithm

SUBMITTED BY

(B-65) Ankur Yadav

(B-67) Uditnarayan Yadav

Submitted in partial fulfilment of the requirements of Operating Systems

UNDER THE GUIDANCE OF

Dr. Amol P. Pande

Department of Computer Engineering

DATTA MEGHE COLLEGE OF ENGINEERING

Sector 3, Airoli, Navi Mumbai-400708

1. Problem statement: To perform LRU(Least Recently Used) Page Replacement Algorithm using C programming Language

2.Objective: In a computer system,that uses Paging for virtual memory management,page replacement algorithms decide which memory pages to page out when a page of memory neends to be allocated.Paging happens when a page fault occurs and free page cannot be used to satisfy the allocation.
When the memory is full, the algorithm must choose Which page to discard to make room for the new ones. There
Are Different types of Algorithms:

- LeastRecentlyUsed(LRU)
- FirstInFirstOut(FIFO)
- MostRecentlyUsed(MRU)
- Random Replacement(RR)

Are Objective in this project, To Replace Page Memory By Using LRU

3.Solution: The solution to these problem is LRU Page Replacement.The LRU paging is to remove the Least recently used frame when the memory is full and a new Page is referenced which is not there in memory.

Access Sequence	5	0	1	2	0	3	1	2	5	2
Frame1	5	5	5	2	2	2	2	2	2	2
Frame 2		0	0	0	0	3	3	3	3	3
Frame 3			1	1	1	1	1	1	5	5
	f	f	f	f		f			f	

4.Features: If the required page is not in the memory, we Bring that in memory.In simple words,The Least Recently Used(LRU)page replacement policy replaces the page that has not been used for the longest period of time.

5.Platform used: C programming language,Windows(Repl Online IDE),Github

6.Source Code:

```
#include<stdio.h>


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;
}

int main()
{
   int no_of_frames, no_of_pages, frames[10], pages[30],
counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;
        printf("Enter number of frames: ");
        scanf("%d", &no_of_frames);

        printf("Enter number of pages: ");
        scanf("%d", &no_of_pages);

        printf("Enter reference string: ");

   for(i = 0; i < no_of_pages; ++i){
        scanf("%d", &pages[i]);
   }

        for(i = 0; i < no_of_frames; ++i){
        frames[i] = -1;
   }
```

```
for(i = 0; i < no_of_pages; ++i){
    flag1 = flag2 = 0;

    for(j = 0; j < no_of_frames; ++j){
        if(frames[j] == pages[i]){
            counter++;
            time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
    }

    if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == -1){
                counter++;
                faults++;
                frames[j] = pages[i];
                time[j] = counter;
                flag2 = 1;
                break;
            }
        }
```

```
        }

        if(flag2 == 0){
                pos = findLRU(time, no_of_frames);
                counter++;
                faults++;
                frames[pos] = pages[i];
                time[pos] = counter;
        }

        printf("\n");

        for(j = 0; j < no_of_frames; ++j){
                printf("%d\t", frames[j]);
        }
        }

        printf("\n\nTotal Page Faults = %d", faults);

    return 0;
}
```

7.Input:

Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 5 7 5 6 7 3

8.Output:

Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 5 7 5 6 7 3

5 -1 -1

5 7 -1

5 7 -1

5 7 6

5 7 6

3 7 6

Total Page Faults = 4

Note:

- (6 to 8 Pages report)
- Text content (Time roman new 12)