Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Summer,Year:2025),B.Sc.in CSE (Day)

LAB REPORT NO - 03

Course Title: Operating System Lab

Course Code: CSE 310 Section: 223-D2

Lab Experiment Name: Implement LFU page replacement algorithm.

Student Details

	Name	ID
1.	MD.SHAJALAL	223002088

Submission Date : 12 – 08 - 2025

Course Teacher's Name : Umme Habiba

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

Objective :

To implement and simulate the **LFU page replacement algorithm** in C, track page faults, and analyze how pages are replaced when the least frequently used page is selected for removal.

Theory :

The **LFU page replacement algorithm** replaces the page that has been used least frequently. It is based on the idea that pages that have been used more often in the past are likely to be used again in the future.

- **Steps**:
    1. Maintain a counter for each page in memory.
    2. When a page is referenced, increment its frequency count.
    3. If a page fault occurs and memory is full, replace the page with the lowest frequency count.
    4. In case of ties (multiple pages with same frequency), the page that entered earliest is replaced (FIFO tie-breaker).

Algorithm :

1. Input **number of frames**, **number of pages**, and **reference string**.
2. Initialize all frames as empty and frequency counts as zero.
3. For each page in the reference string:
    - If the page exists in memory → no page fault (just update frequency).
    - If not:
        - If a free frame exists → insert the page and set frequency = 1.
        - Else → find the page with the **lowest frequency**, replace it with the new page.
4. Count total page faults.
5. Display the page replacement process and total page faults.

C Program :

```c
#include <stdio.h>
#define MAX_FRAMES 10
#define MAX_PAGES 50
int main() {
    int frames[MAX_FRAMES], freq[MAX_FRAMES], time[MAX_FRAMES];
    int nFrames, nPages, pages[MAX_PAGES];
    int i, j, k, minFreq, minTime, pageFaults = 0, found, pos, counter = 0;
    printf("Enter number of frames: ");
    scanf("%d", &nFrames);
    printf("Enter number of pages: ");
    scanf("%d", &nPages);
    printf("Enter reference string: ");
    for(i = 0; i < nPages; i++) {
        scanf("%d", &pages[i]);
    }
    for(i = 0; i < nFrames; i++) {
        frames[i] = -1;
        freq[i] = 0;
        time[i] = 0;
    }
    printf("\nThe Page Replacement Process is ->\n");
    for(i = 0; i < nPages; i++) {
        found = 0;
        for(j = 0; j < nFrames; j++) {
            if(frames[j] == pages[i]) {
                freq[j]++;
                found = 1;
                break;
            }
        }
        if(found) {
            printf("For %d : No page fault!\n", pages[i]);
        } else {
            pageFaults++;
            for(j = 0; j < nFrames; j++) {
                if(frames[j] == -1) {
                    frames[j] = pages[i];
                    freq[j] = 1;
                    time[j] = counter++;
                    found = 1;
                    break;
                }
```

```
            }
            if(!found) {
                minFreq = freq[0];
                pos = 0;
                for(j = 1; j < nFrames; j++) {
                    if(freq[j] < minFreq || (freq[j] == minFreq && time[j] <
time[pos])) {
                        minFreq = freq[j];
                        pos = j;
                    }
                }
                frames[pos] = pages[i];
                freq[pos] = 1;
                time[pos] = counter++;
            }
            printf("For %d : ", pages[i]);
            for(k = 0; k < nFrames; k++) {
                if(frames[k] != -1)
                    printf("%d ", frames[k]);
            }
            printf("\n");
        }
    }
    printf("\nTotal no of page faults using LFU is: %d\n", pageFaults);
    return 0;
}
```

Output:

```
PS D:\7th Semester\Operating System Lab\Lab-Reort-03\output> & .\'Lab-Reort-03.exe'
Enter number of frames: 3
Enter number of pages: 20
Enter reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

The Page Replacement Process is ->
For 7 : 7
For 0 : 7 0
For 1 : 7 0 1
For 2 : 2 0 1
For 0 : No page fault!
For 3 : 2 0 3
For 0 : No page fault!
For 4 : 4 0 3
For 2 : 4 0 2
For 3 : 3 0 2
For 0 : No page fault!
For 3 : No page fault!
For 2 : No page fault!
For 1 : 3 0 1
For 2 : 3 0 2
For 0 : No page fault!
For 1 : 3 0 1
For 7 : 3 0 7
For 0 : No page fault!
For 1 : 3 0 1

Total no of page faults using LFU is: 13
```

Conclusion :

The LFU page replacement algorithm effectively reduces the number of page faults by replacing the least frequently used pages. For the given reference string and 3 frames, the total number of page faults was 13. This approach works well when past usage frequency is a good predictor of future use, but may perform poorly if page access patterns change suddenly.