EXPERIMENT NO. 08

LRU PAGE REPLACEMENT ALGORITHM

Objective

To write a program to simulate the Least Recently Used (LRU) page replacement algorithm.

Theory

The Least Recently Used (LRU) page replacement algorithm replaces the page that has not been used for the longest time. It uses a priority system to determine which page to replace when the memory is full. This algorithm provides better performance than FIFO by considering usage history.

Algorithm Steps

1. Initialize an empty memory frame array.
2. Traverse each page in the reference string.
3. If the page exists in memory, mark it as recently used.
4. If the page does not exist and memory is full, remove the least recently used page.
5. Insert the current page into memory and update recent usage.
6. Count the number of page faults encountered.

Program Code (C Language)

```c
#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;
   int time[10], flag1, flag2, i, j, pos, faults = 0;

   printf("Enter number of pages: ");
   scanf("%d", &no_of_pages);
   printf("Enter page reference string: ");
   for(i = 0; i < no_of_pages; ++i) {
      scanf("%d", &pages[i]);
   }

   printf("Enter number of frames: ");
   scanf("%d", &no_of_frames);

   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("Frames after page %d: ", pages[i]);
    for(j = 0; j < no_of_frames; ++j) {
        if(frames[j] != -1)
            printf("%d ", frames[j]);
    }
    printf("\n");
  }

  printf("\nTotal Page Faults = %d\n", faults);
  return 0;
}
```

Sample Output

Enter number of pages: 12

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

Enter number of frames: 4

Frames after page 1: 1

Frames after page 2: 1 2

Frames after page 3: 1 2 3

Frames after page 4: 1 2 3 4

Frames after page 1: 1 2 3 4

Frames after page 2: 1 2 3 4

Frames after page 5: 1 2 5 4

Frames after page 1: 1 2 5 4

Frames after page 2: 1 2 5 4

Frames after page 3: 1 2 5 3

Frames after page 4: 1 2 4 3

Frames after page 5: 5 2 4 3

Total Page Faults = 8

Conclusion

The LRU page replacement algorithm improves memory usage by removing the least recently accessed page. It provides better efficiency than FIFO in most practical scenarios.
