A Mini Project Report

on

Page Replacement Algorithm: Least Recently Used

In Subject: ADUA21203: Microprocessors

by

Shardul Khandebharad (272026)
Neemeesh Khanzode (272027)
Jyotirmay Khavasi (272028)
Suraj Raskar (272046)

Department of Artificial Intelligence and Data Science

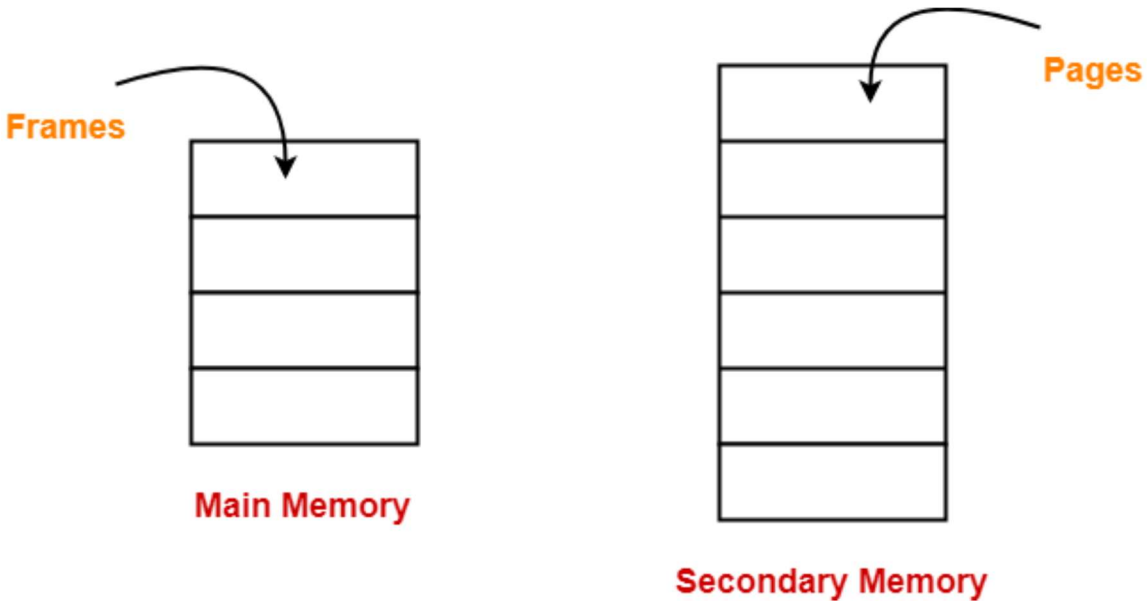
VIIT

2020-2021

Contents

Introduction: Our project is based on Development on Replacement Algorithm-LRU

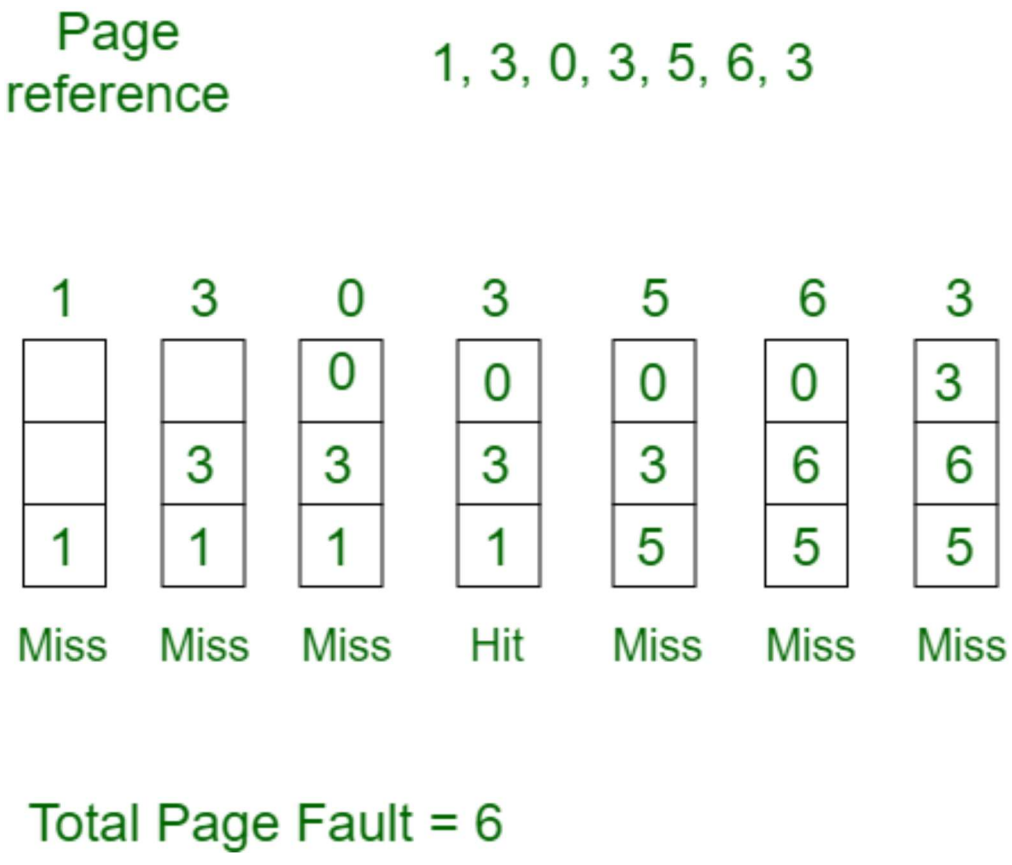
- **Paging:**
  - Paging is a fixed size partitioning scheme.
  - In paging, secondary memory and main memory are divided into equal fixed size partitions.
  - The partitions of secondary memory are called as **pages**.
  - The partitions of main memory are called as **frames**.



- **Page Fault**
  - **When a page referenced by the CPU is not found in the main memory, it is called as a page fault.**
  - **When a page fault occurs, the required page has to be fetched from the secondary memory into the main memory**

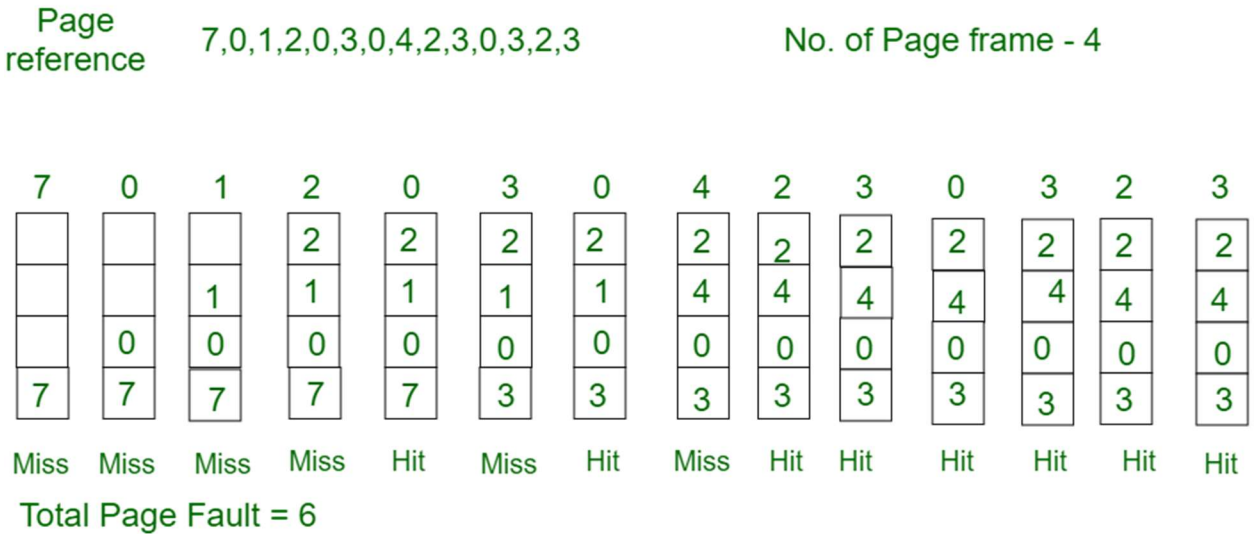
Page Replacement Algorithms:

- FIFO page replacement algorithm:
  - FIFO - **First In First Out**
  - 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.



- LIFO page replacement algorithm
  - Works on the principle of **Last in First out**.
  - It replaces the newest page that arrived at last in the main memory.
  - It is implemented by keeping track of all the pages in a stack.

- Optimal page replacement algorithm
  - In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.



- Random page replacement algorithm
  - Replaces the page randomly.
  - This Algorithm can work like any other page replacement algorithm that is LIFO, FIFO, Optimal, and LRU.

- **LRU Page Replacement Algorithm**
  - In the Least Recently Used (LRU) page replacement policy, **the page that is used least recently will be replaced**.
  - If a process requests for page and that page is found in the main memory then it is called **page hit**, otherwise **page miss or page fault**

    Advantages of the Least Recently Used

    Least recently used memory page replacement algorithm does not suffer from Belady's anomaly. In using first-in-first-out (FIFO) page replacement algorithm, there is the occurrence of Belady's anomaly a phenomenon in which an increase in the number of page frames translates to an increase in the number of page faults for certain memory access patterns. Another advantage of LRU is that it is relatively easy to implement as it is not very complex, relies on simple data structures and facilitates ease in page replacement. Least Recently Used page algorithms also facilitate optimal page replacement which has the least rate of occurrence of page faults which if too much affects the memory capacity.

Disadvantages of the Least Recently Used

Although the Least Recently Used page algorithms also facilitate optimal page replacement the process can be time consuming. Another disadvantage is that error detection is difficult compared to using first-in-first-out (FIFO) page replacement algorithms. Also LRUS have limited applicability especially as not operating systems can implement the algorithm. LRUS are also very expensive to operate and call for technical expertise in their usage further adding to their operational costs as skilled manpower is often expensive to acquire and maintain. LRU also suffers from difficulties in its basic form and the problem is that there are inconsistencies in page accesses, that is a page can easily and heavily be accessed for a second and remain unavailable for up to the next 10 seconds. In addition, Least Recently Used page algorithms are affected by the need for hardware support and that paging significantly slows down memory access because of large page sizes.

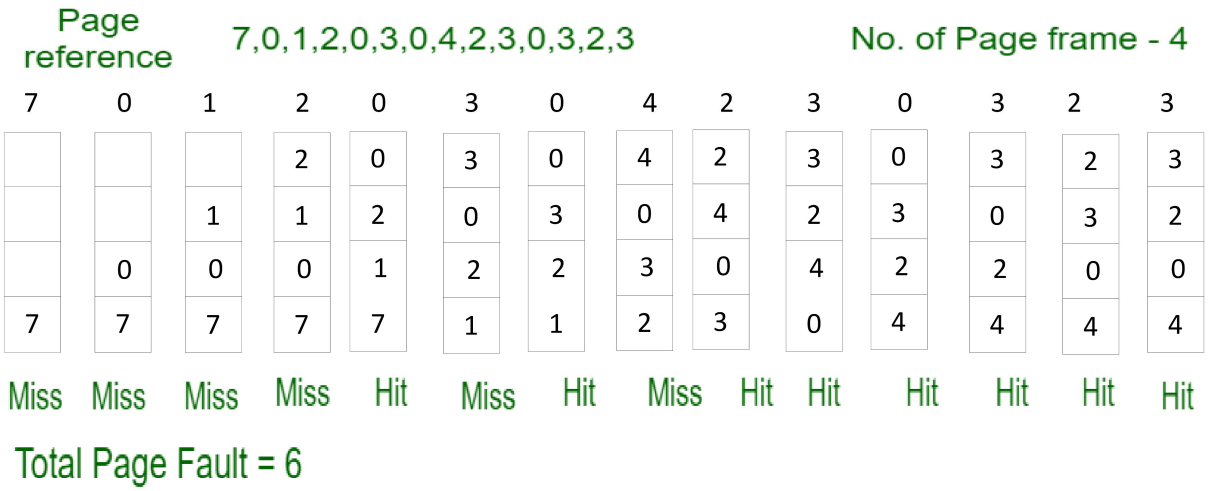
Requirements

- C/C++ Compiler like Vs code / GDB online compiler / Inteli J etc.

Problem statement

- To Develop a Least Recently Used algorithm in C/C++.

Example-



Here LRU has same number of page fault as optimal but it may differ according to question.

Code-

```
#include <iostream>
#include<deque>
#include <algorithm>
#include <conio.h>
#include <stdlib.h>
using namespace std;
void showdq(deque <int> g){

   deque <int> :: iterator it;
   for (it = g.begin(); it != g.end(); ++it) cout << '\t' << *it;
   cout << '\n';
}
void HitMissCalculate(int capacity,int size,int arr[]){
   deque<int> q(capacity);
   int count=0;
   int temp=0;
   int miss=0,hit=0;
   deque<int>::iterator itr;
```

```
   q.clear();

   //insert next page into the queue if not present in queue and inc miss
   for(int i=0;i<size;i++){
      itr = find(q.begin(),q.end(),arr[i]);//find whether new page is present in the queue
and stores it in itr and if not found it stores q.end
      if(itr == q.end()){
         miss++;
         if(q.size() == capacity){     //checks whether frame in full or not
            q.erase(q.begin());        //deletes LRU page
            q.push_back(arr[i]);       //Push new page to MRU
         }
         else q.push_back(arr[i]);      //push new page to MRU
      }
      else{
         hit++;
         q.erase(itr);   //deletes found page
         q.push_back(arr[i]);               //push the same page to MRU
      }
      //For printing
      if(temp==0){
      cout<<"\tLRU";
      for(int i=0;i<capacity-1;i++) cout<<"\t";
      cout<<"MRU\n";
      temp=1;}
      showdq(q);
   }
   cout<<"\nFor frame size "<<capacity<<" and having pages ";
   for(int i=0;i<size;i++) cout<<arr[i]<<" ";
   cout<<"\nNo. of page faults(miss): "<<miss;
   cout<<"\nNo. of page hits are: "<<hit;
   float hratio=hit*100/(hit+miss), mratio=miss*100/(hit+miss);
   cout<<"\nHit %: "<<hratio;
   cout<<"\nMiss %: "<<mratio;
}
int main(){
   int choice;
   cout<<"***Least Reacently Used Implementation***";
      int capacity,size;
      cout<<"\nEnter Frame size: ";
      cin>>capacity;
      cout<<"\nEnter no. of pages: ";
      cin>>size;
      cout<<"\nEnter the pages seperated by spaces:\n";
```

```
    int arr[size];
    for(int i=0;i<size;i++) cin>>arr[i];
    HitMissCalculate(capacity,size,arr);
  while(1){
    cout<<"\n1. Find hit/miss for the same pages with different framesize.\n2. Apply
LRU for all different inputs.\n3. Exit.\nEnter your choice: ";
    cin>>choice;
    switch(choice){
      case 1:
        cout<<"\nEnter Frame size: ";
        cin>>capacity;
        HitMissCalculate(capacity,size,arr);
        break;
      case 2:
        system("cls");
        cout<<"***Least Reacently Used Implementation***";
        cout<<"\nEnter Frame size: ";
        cin>>capacity;
        cout<<"\nEnter no. of pages: ";
        cin>>size;\
        cout<<"\nEnter the pages seperated by spaces:\n";
        for(int i=0;i<size;i++) cin>>arr[i];
        HitMissCalculate(capacity,size,arr);
        break;
      case 3:
        cout<<"Exiting.....";
        return 0;
      default:
      cout<<"Enter the valid choice."<<endl;
      break;
    } } }
```

Optimal Use Case & Output:

```
***Least Reacently Used Implementation***
Enter Frame size: 8

Enter no. of pages: 40

Enter the pages seperated by spaces:
0 2 0 2 7 9 2 0 0 8 5 2 2 0 0 7 6 5 4 1 2 0 0 2 0 5 0 1 2 0 0 3 0 9 1 0 2 0 0 1
        LRU                                                             MRU
        0
        0       2
        2       0
        0       2
        0       2       7
        0       2       7       9
        0       7       9       2
        7       9       2       0
        7       9       2       0
        7       9       2       0       8
        7       9       2       0       8       5
        7       9       0       8       5       2
        7       9       0       8       5       2
        7       9       8       5       2       0
        7       9       8       5       2       0
        9       8       5       2       0       7
        9       8       5       2       0       7       6
        9       8       2       0       7       6       5
        9       8       2       0       7       6       5       4
        8       2       0       7       6       5       4       1
        8       0       7       6       5       4       1       2
        8       7       6       5       4       1       2       0
        8       7       6       5       4       1       2       0
        8       7       6       5       4       1       0       2
        8       7       6       5       4       1       2       0
        8       7       6       4       1       2       0       5
        8       7       6       4       1       2       5       0
        8       7       6       4       2       5       0       1
        8       7       6       4       5       0       1       2
        8       7       6       4       5       1       2       0
        8       7       6       4       5       1       2       0
        7       6       4       5       1       2       0       3
        7       6       4       5       1       2       3       0
        6       4       5       1       2       3       0       9
        6       4       5       2       3       0       9       1
        6       4       5       2       3       9       1       0
        6       4       5       3       9       1       0       2
        6       4       5       3       9       1       2       0
        6       4       5       3       9       1       2       0
        6       4       5       3       9       2       0       1

For frame size 8 and having pages 0 2 0 2 7 9 2 0 0 8 5 2 2 0 0 7 6 5 4 1 2 0 0 2 0 5 0 1 2 0 0 3 0 9 1 0 2
0 0 1
No. of page faults(miss): 11
No. of page hits are: 29
Hit %: 72
Miss %: 27
1. Find hit/miss for the same pages with different framesize.
2. Apply LRU for all different inputs.
3. Exit.
Enter your choice:
```

```
1. Find hit/miss for the same pages with different framesize.
2. Apply LRU for all different inputs.
3. Exit.
Enter your choice: 1

Enter Frame size: 5
        LRU                             MRU
        0
        0       2
        2       0
        0       2
        0       2       7
        0       2       7       9
        0       7       9       2
        7       9       2       0
        7       9       2       0
        7       9       2       0       8
        9       2       0       8       5
        9       0       8       5       2
        9       0       8       5       2
        9       8       5       2       0
        9       8       5       2       0
        8       5       2       0       7
        5       2       0       7       6
        2       0       7       6       5
        0       7       6       5       4
        7       6       5       4       1
        6       5       4       1       2
        5       4       1       2       0
        5       4       1       2       0
        5       4       1       0       2
        5       4       1       2       0
        4       1       2       0       5
        4       1       2       5       0
        4       2       5       0       1
        4       5       0       1       2
        4       5       1       2       0
        4       5       1       2       0
        5       1       2       0       3
        5       1       2       3       0
        1       2       3       0       9
        2       3       0       9       1
        2       3       9       1       0
        3       9       1       0       2
        3       9       1       2       0
        3       9       1       2       0
        3       9       2       0       1

For frame size 5 and having pages 0 2 0 2 7 9 2 0 0 8 5 2 2 0 0 7 6 5 4 1 2 0 0 2 0 5 0 1 2 0 0 3 0 9 1
0 2 0 0 1
No. of page faults(miss): 14
No. of page hits are: 26
Hit %: 65
Miss %: 35
1. Find hit/miss for the same pages with different framesize.
2. Apply LRU for all different inputs.
3. Exit.
Enter your choice:
```

Conclusion:

In this project we developed LRU algo with the help of C/C++.As well as we studied about Page Replacement Algorithms. And we also get a application of LRU.

Application:

SAMSUNG'S TOUCHWIZ

SHOWS RECENTLY USED APPS ON HOME SCREEN

Samsung touch wiz was a user interface of Samsung mobiles which is based on android. It was seen in mostly galaxy series of Samsung company.
In Samsung's touch wiz user interface there were a feature where it shows most recently used app on home screen of mobile.
As Home Screen has 12 slots. If we open a new app which is not among this 12 apps it will automatically removes least recently used app add the new used app in one of the 12 slots.
This feature purely inspired by LRU Algo. which was very time saving feature.

Operations:

- Keep the most recently used apps at the front of the list.
- When the user opens an app not in the list, add this app to the front of the list.
- If the list is full, remove the least recently used app from the list

References

https://en.wikipedia.org/wiki/Cache_replacement_policies

https://www.interviewcake.com/concept/java/lru-cache

https://www.geeksforgeeks.org/deque-set-1-introduction-applications/