DOP: 21 OCT, 2021 31443.

Assignment No: 07 (Group B): Page Replacement Algorithm.

1) Title: Write a program to simulate page replacement algorithm.

2) Software and hardware requirement:

* Software(s)

1) Integrated Development Enviroment :- CLion.
2) Cpp Compiler.
3) (OR) Notepad.

* Hardware

1) Computer System :-
a) Processor : i5 gen
b) Ram : 8 GB
c) Monitor : 1080p | 720p FHD | IPS.
d) I/O Peripherals
- Keyboard.
- Mouse.

3) Learning Objective:

1) Understand what is page replacement Algorithm.

2) Understand what is demand paging.
3) Understand page replacement algorithm and its working.

4) Learning Outcome:

1) One will be able to understand the concept of page replacement, its algorithms, and what is demand paging.

2) One will be able to implement page replacement algorithms using appropriate data structure set and programming skill set.

5) Concept Related Theory

Demand Paging: According to the concept of virtual memory, in order to execute some process, only a part of the process needs to be present in the main memory which means that only a few pages will only be present in the main memory at any time.

However, deciding which page need to be kept in the main memory and which need to be kept in the secondary memory, is going to be difficult because we cannot say in advance that a pr-

will require a particular page at particular time.

Therefore, to overcome this problem, there is a concept called demand paging is introduced. it suggests keeping all pages of the frame in the secondary memory until they are required. In other words, it says that do not load any page in the main memory until it is required.

Whenever any page is referred for the first time in the main memory, then that page will be found in the secondary memory.

After that, it may or may not be present in the main memory depending upon the page replacement algorithm which will be covered later in this tutorial.

* Page Replacement Algorithm in OS.

In operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new pages comes in.

Page fault : A page fault happens when a running program access a memory page that is mapped into the virtual address space, but ~~not~~ not loaded in physical memory.

Since, actual memory is much smaller than virtual memory, page fault happens. In case of page fault, OS might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace the target for all algorithms is to reduce the number of page faults.

1) First-in-First-Out [FIFO]:

The OS 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 queue is selected for removal.

Example:

Pages: 1 3 0 3 5 6 3

No of frames : 03.

Page String	1	3	0	3	5	6	3
	0	-	-	0	0	0	0	3
	1	-	3	3	3	3	6	6
	2	1	1	1	1	5	5	5

Page faults = 6.

Belady's Anomaly : Belady's Anomaly proves that it is possible to have more page faults when increasing the ~~Add~~ number of page frames while using this algo.

2) Optimal Page Replacement Algorithm.

In this Algorithm, pages are replaced which would not be used for the longest duration of time in future.

Example :-

Page String :	7	0	1	2	0	3	0	4	2	3	0	3	2	3
Frames 0	-	-	-	2	2	2	2	2	2	2	2	2	2	2
1	-	-	1	1	1	1	1	4	4	4	4	4	4	4
2	-	0	0	0	0	0	0	0	0	0	0	0	0	0
3	7	7	7	7	7	3	3	3	3	3	3	3	3	3

Page faults : 6

3) Least Recently Used:

In this algorithm page will be replaced which is least recently used.

Example:

Reference String	7	0	1	2	0	3	0	4	2	3	0	3	2	3
frames	0	-	-	-	2	2	2	2	2	2	2	2	2	2
	1	-	-	1	1	1	1	1	4	4	4	4	4	4
	2	-	0	0	0	0	0	0	0	0	0	0	0	0
	3	7	7	7	7	7	3	3	3	3	3	3	3	3

Page faults : 6.

6) Conclusion :

Understood and implemented the concept of page replacement algorithms.

7) References

1) GeeksForGeeks.
2) JavaTpoint.

Code:

```
#include<iostream>
using namespace std;

int *initializeFrames(int frames[],int numOfFrames){
    for(int i=0;i<numOfFrames;i++){
        frames[i] = -1;
    }
    return frames;
}

//Supporting functions for LRU starts
int present(int table_frame[], int nf, int page)
{
    for(int i=0; i<nf; i++)
        if(page == table_frame[i])
            return 1;
    return 0;
}

void printtable(int table_frame[], int nf)
{
    for(int i=0; i<nf; i++)
    {
        if(table_frame[i] == -1)
            printf(" -\t");
        else
            printf("%2d\t", table_frame[i]);
    }
}

int findpos(int table_frame[], int nf, int pages[], int curr, int np)
{
    for(int i=0; i<nf; i++)
        if(table_frame[i] == -1)
            return i;

    int pos[nf] = {0};
    for(int i=0; i<nf; i++)
    {
        pos[i] = -1e9;
        for(int j=curr-1; j>=0; j--)
            if(pages[j] == table_frame[i])
            {
                pos[i] = j;
                break;
            }
    }

    int min1 = 1000000, retPos = -1;
    for(int i=0; i<nf; i++)
        if(min1 > pos[i])
        {
            min1 = pos[i];
            retPos = i;
        }
```

```
    return retPos;
}
//LRU Functions ENDS
void algoLRU(int numOfPages,int numOfFrames,int pageString[],int frames[]){
    bool pagePresentStatus;
    int pageFault, pageHit;
    frames = initializeFrames(frames,numOfFrames);
    /*for(int i=0;i<numOfPages;i++){
        for(int j=0;j<numOfFrames;j++)
        {
            if(pageString[i] == frames[j]) pagePresentStatus = true;
        }
        if(!pagePresentStatus){
            pageFault++;
        } else{
            pageHit++;
        }
        cout<<endl;
        for(int k=0;k<numOfFrames;k++){
            cout<<frames[k]<<" ";
        }
        pagePresentStatus = false;
    }
    cout<<"\n\nPage Faults : "<<pageFault<<endl;
    cout<<"Page Hits : "<<pageHit<<endl;*/
    int n=numOfPages,nf=numOfFrames,i,pos=0;
    pageFault=0;
    pageHit = 0;
    cout<<"\n**Page Replacement using Least Recently Used
Algorithm**\n"<<endl;
    for(i=0;i<n;i++)
    {
        //printf("Page %2d ->\t",pageString[i]);
        cout<<"Page "<<pageString[i]<<"\t->\t";
        if(!present(frames,nf,pageString[i]))
        {
            int pos = findpos(frames,nf,pageString,i,n);
            frames[pos]=pageString[i];

            printtable(frames,nf);
            printf("\'F\'\n");
            pageFault++;
            continue;
        }else{
            printtable(frames,nf);
            printf("\'H\'\n");
            pageHit++;
        }
    }
    printf("\nPage faults : %d", pageFault);
    printf("\nPage Hit : %d\n\n", pageHit);

}
void algoOptimal(int numOfPages,int numOfFrames,int pageString[],int
frames[]){
    int temp[10], flag1, flag2, flag3, i, j, k, pos, max,
```

```
pageFaults=0,pageHits=0;
    bool presentStatus;
    initializeFrames(frames,numOfFrames);
    cout<<endl<<"**Page Replacement using Optimal Algorithm**"<<endl;
    for(i = 0; i < numOfPages; ++i){
        flag1 = flag2 = 0;
        presentStatus = false;
        for(j = 0; j < numOfFrames; ++j){
            if(frames[j] == pageString[i]){
                flag1 = flag2 = 1;
                pageHits++;
                presentStatus = true;
                break;
            }
        }

        if(flag1 == 0){
            for(j = 0; j < numOfFrames; ++j){
                if(frames[j] == -1){
                    pageFaults++;
                    frames[j] = pageString[i];
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
            flag3 =0;

            for(j = 0; j < numOfFrames; ++j){
                temp[j] = -1;

                for(k = i + 1; k < numOfPages; ++k){
                    if(frames[j] == pageString[k]){
                        temp[j] = k;
                        break;
                    }
                }
            }

            for(j = 0; j < numOfFrames; ++j){
                if(temp[j] == -1){
                    pos = j;
                    flag3 = 1;
                    break;
                }
            }

            if(flag3 ==0){
                max = temp[0];
                pos = 0;

                for(j = 1; j < numOfFrames; ++j){
                    if(temp[j] > max){
                        max = temp[j];
                        pos = j;
```

```
                    }
                }
            }

            frames[pos] = pageString[i];
            pageFaults++;
        }

        cout<<endl;
        cout<<"Page "<<pageString[i]<<"\t->\t";
        for(j = 0; j < numOfFrames; ++j){
            (frames[j]==-1)?cout<<"-"<<"\t":cout<<frames[j]<<"\t";
        }
        presentStatus?cout <<"\'H\'":cout <<"\'F\'";
    }

    cout<<"\n\nPage Faults : "<<pageFaults<<endl;
    cout<<"Page Hits : "<<pageHits<<endl<<endl;
}
void algoFIFO(int numOfPages,int numOfFrames,int pageString[],int frames[]){
    /*frames = initializeFrames(frames,numOfFrames);
    cout<<"num of pages : "<<numOfPages<<endl;
    cout<<"num of Frames : "<<numOfFrames<<endl;
    cout<<"String : ";
    for(int i =0;i<numOfPages;i++){
        cout<<" "<<pageString[i];
    }
    cout<<"\nFrames : ";
    for(int i =0;i<numOfFrames;i++){
        cout<<" "<<frames[i];
    }
    cout<<endl;*/
    int cachePosPointer = 0;
    bool presentStatus;
    int pageFault =0, pageHit = 0;
    frames = initializeFrames(frames,numOfFrames);
    cout<<endl<<"**Page Replacement using FIFO Algorithm**"<<endl;
    for(int i=0;i<numOfPages;i++){
        for(int j=0;j<numOfFrames;j++)
        {
            if(pageString[i] == frames[j]) presentStatus = true;
        }
        if(!presentStatus){
            if(cachePosPointer==numOfFrames) cachePosPointer = 0;
            frames[cachePosPointer++]=pageString[i];
            pageFault++;
        } else{
            pageHit++;
        }
        cout<<endl;
        cout<<"Page "<<pageString[i]<<"\t->\t";
        for(int k=0;k<numOfFrames;k++){
            frames[k]==-1?cout<<"-"<<"\t":cout<<frames[k]<<"\t";
        }
        presentStatus?cout <<"\'H\'":cout <<"\'F\'";
        presentStatus = false;
    }
```

```
    cout<<"\n\nPage Faults : "<<pageFault<<endl;
    cout<<"Page Hits : "<<pageHit<<endl<<endl;
}

int main(){
    int numOfPages,numOfFrames,choice;
    cout<<"Enter number of Pages : ";
    cin>>numOfPages;
    int pages[numOfPages];
    cout<<endl<<"Enter number of Frames : ";
    cin>>numOfFrames;
    int frames[numOfFrames];
    cout<<"\nEnter Pages ->\n";
    for(int r=0;r<numOfPages;r++){
        cout<<"Enter page "<<(r+1)<<" : ";
        cin>>pages[r];
    }
    while(1) {
        cout<<"|------------------------|\n";
        cout<<"| Menu                   |\n|------------------------|"
              "\n| 1: Least Recently Used |\n| 2: Optimal             |\n| 3:
First In First Out  |\n| 4: Exit                |\n|------------------------
|\nEnter your choice : ";
                cin>>choice;
        switch (choice) {
            case 1:
                algoLRU(numOfPages,numOfFrames,pages,frames);
                break;
            case 2:
                algoOptimal(numOfPages,numOfFrames,pages,frames);
                break;
            case 3:
                algoFIFO(numOfPages,numOfFrames,pages,frames);
                break;
            case 4:
                cout<<endl<<"Terminated..!";
                exit(0);
                break;
            default:
                cout << "\nPlease Enter valid choice between 1 to 4\n";
        }
    }
    return 0;

}
```

Output:

```
Enter number of Pages :14

Enter number of Frames :4

Enter Pages ->
Enter page 1 :7
Enter page 2 :0
Enter page 3 :1
Enter page 4 :2
Enter page 5 :0
Enter page 6 :3
Enter page 7 :0
Enter page 8 :4
Enter page 9 :2
Enter page 10 :3
Enter page 11 :0
Enter page 12 :3
Enter page 13 :2
Enter page 14 :3
|--------------------------------|
| Menu                           |
|--------------------------------|
| 1: Least Recently Used         |
| 2: Optimal                     |
| 3: First In First Out          |
| 4: Exit                        |
|--------------------------------|
Enter your choice :1

**Page Replacement using Least Recently Used Algorithm**

Page 7  ->   7   -   -   -   'F'
Page 0  ->   7   0   -   -   'F'
Page 1  ->   7   0   1   -   'F'
Page 2  ->   7   0   1   2   'F'
Page 0  ->   7   0   1   2   'H'
Page 3  ->   3   0   1   2   'F'
Page 0  ->   3   0   1   2   'H'
Page 4  ->   3   0   4   2   'F'
Page 2  ->   3   0   4   2   'H'
Page 3  ->   3   0   4   2   'H'
Page 0  ->   3   0   4   2   'H'
Page 3  ->   3   0   4   2   'H'
```

```
Page 2  ->      3      0      4      2     'H'
Page 3  ->      3      0      4      2     'H'

Page faults : 6
Page Hit : 8

|--------------------------------|
| Menu                           |
|--------------------------------|
| 1: Least Recently Used  |
| 2: Optimal                     |
| 3: First In First Out          |
| 4: Exit                        |
|--------------------------------|
Enter your choice :2

**Page Replacement using Optimal Algorithm**

Page 7  ->      7      -      -      -     'F'
Page 0  ->      7      0      -      -     'F'
Page 1  ->      7      0      1      -     'F'
Page 2  ->      7      0      1      2     'F'
Page 0  ->      7      0      1      2     'H'
Page 3  ->      3      0      1      2     'F'
Page 0  ->      3      0      1      2     'H'
Page 4  ->      3      0      4      2     'F'
Page 2  ->      3      0      4      2     'H'
Page 3  ->      3      0      4      2     'H'
Page 0  ->      3      0      4      2     'H'
Page 3  ->      3      0      4      2     'H'
Page 2  ->      3      0      4      2     'H'
Page 3  ->      3      0      4      2     'H'

Page Faults : 6
Page Hits : 8

|--------------------------------|
| Menu                           |
|--------------------------------|
| 1: Least Recently Used  |
| 2: Optimal                     |
| 3: First In First Out          |
| 4: Exit                        |
|--------------------------------|
```

```
Enter your choice :3

**Page Replacement using FIFO Algorithm**

Page 7  ->    7     -     -     -    'F'
Page 0  ->    7     0     -     -    'F'
Page 1  ->    7     0     1     -    'F'
Page 2  ->    7     0     1     2    'F'
Page 0  ->    7     0     1     2    'H'
Page 3  ->    3     0     1     2    'F'
Page 0  ->    3     0     1     2    'H'
Page 4  ->    3     4     1     2    'F'
Page 2  ->    3     4     1     2    'H'
Page 3  ->    3     4     1     2    'H'
Page 0  ->    3     4     0     2    'F'
Page 3  ->    3     4     0     2    'H'
Page 2  ->    3     4     0     2    'H'
Page 3  ->    3     4     0     2    'H'

Page Faults : 7
Page Hits : 7

|--------------------------------|
| Menu                           |
|--------------------------------|
| 1: Least Recently Used         |
| 2: Optimal                     |
| 3: First In First Out          |
| 4: Exit                        |
|--------------------------------|
Enter your choice :4

Terminated..!
Process finished with exit code 0
```