**Don Bosco Institute of Technology,Kurla**
**Academic Year 2019-20**

**EXPERIMENT NO: 6**

# Title: A program to simulate cache memory management using page replacement algorithms

**By ~**

**Shawn Louis**

**Roll no : 31**

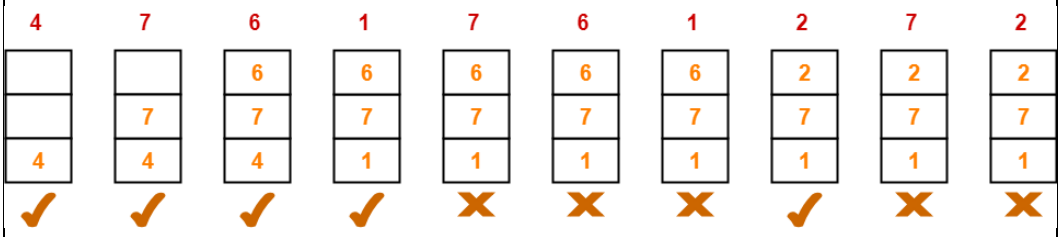
**Class:S.E Comps(Sem IV)** **Lecturer:Sejal.Chopra**

**Subject: PA Lab**

## EXPERIMENT NO:6

## Page replacement algorithms

| | |
|---|---|
| **AIM** | Write a program to simulate cache memory management usingpage replacement algorithms. |
| **LEARNING OBJECTIVE** | To implement various page replacement policies. |
| **LEARNING OUTCOME** | Studentwillbeabletovisualizethescenariowhennewpagesenterthe cache memory using various algorithm. |
| **LAB OUTCOME** | CSL 403.1: Ability to compile a code for computer operations. |
| **PROGRAM OUTCOME** | PO1•1,<br>PO5•2,<br>PO8•3,<br>PO9•3,<br>PO12•2,<br>PSO1•2 |
| **BLOOM'S TAXONOMY LEVEL** | Remember,<br>Understand |
| **THEORY** | In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page.<br><br>**L**east **R**ecently **U**sed (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used.<br>A good approximation to the optimal algorithm is based on the observation that pages that have been heavily used in the last few instructions will probably be heavily used again in the next few. Conversely, pages that have not been used for ages will probably remain unused for a long time. This idea suggests a realizable algorithm: when a page fault occurs, throw out the page that has been unused for the longest time. This strategy is called **LRU** (**Least Recently Used**) paging.<br><br>Advantage:<br>The advantage of LRU page replacement algorithm is that it does not suffer from Belady's anomaly which is the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given |

| | |
|---|---|
| | memory access pattern because they belong to a class of stack based page replacement algorithms.<br><br>Disadvantage:<br>Although LRU is theoretically realizable, it is not cheap. To fully implement LRU, it is necessary to maintain a linked list of all pages in memory, with the most recently used page at the front and the least recently used page at the rear. The difficulty is that the list must be updated on every memory reference. Finding a page in the list, deleting it, and then moving it to the front is a very time consuming operation, even in hardware (assuming that such hardware could be built).<br><br>Example:<br>  |
| **SOFTWARE USED** | C/C++/Java |

| | |
|---|---|
| **STEPS TO EXECUTE THE PROGRAM** | 1.Ask the user to enter the frame size.(ex:take it 3) 2.Let him enter the number of pages.<br>3. Ask the user to enter the page numbers(reference string).<br>4. Initially there occurs three(same as your frame size) page faults while filling the frame.<br>5. After that when the frame is full, the page is replaced depending on the specific page replacement algorithm.<br>6. Whenever the same page appears in the frame ,a hit occurs.<br>7.Display in each clock cycle the contents of the frame. ie the page numbers and show whether it is a hit or a miss.<br>8.Calculate the total no.of hits.misses and the hit ratio(no.of hits/total number of pages entered) and miss ratio or fault ratio (no.of misses/total number of pages entered). |

**CODE**

```
#include<stdio.h>

int LRU(int time[], int n);

int main()
{
    int nof, nog, f[10], p[30], c = 0, t[10], flag1, flag2, i, j, pos, faults = 0,hit=0;
        float x1,x2;
        printf("Enter number of f: ");
        scanf("%d", &nof);

        printf("Enter number of p: ");
        scanf("%d", &nog);
        int n=nog;
        printf("Enter the page number: ");

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

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

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

    for(j = 0; j < nof; ++j){ if(f[j] == p[i]){
                    c++; hit++;
                    t[j] = c;
                        flag1 = flag2 = 1;
                        break;
                    }
    }

    if(flag1 == 0){
                    for(j = 0; j < nof; ++j){ if(f[j] == -1){
                        c++;
                        faults++;
                        f[j] = p[i]; t[j] = c; flag2
                        = 1;
                        break;
                    }
            }
    }

    if(flag2 == 0){
            pos = LRU(t, nof); c++;
            faults++;
            f[pos] = p[i]; t[pos] = c;
    }

    printf("\n");
    printf(" ---------------------------------------------- \n");
    for(j = 0; j < nof; ++j){ printf("%d\t",
            f[j]);
    }
            if (flag1==1 && flag2==1)
            printf("\tHIT");
            else
            printf("\tMISS");
    }
x1= (float)hit/n; x2=
(float)faults/n;
    printf("\n\nTotal MISS = %d", faults);
    printf("\n\nTotal HITS = %d",hit);
    printf("\n\nMISS ratio = %.2f",x2);
    printf("\n\nHIT ratio = %.2f",x1);
```

```
    return 0;
}


int LRU(int time[], int n){
        int i, min = t[0], pos = 0;

        for(i = 1; i < n; ++i){
                if(t[i] < min){ min = t[i]; pos =
                        i;
                }
        }

        return pos;
}
```

**OUTPUT**

```
Enter number of frames: 3
Enter number of pages: 12
Enter the page number: 2
3
2
1
5
2
4
5
3
2
5
2

----------------------------------------
2       -1      -1              MISS
----------------------------------------
2       3       -1              MISS
----------------------------------------
2       3       -1              HIT
----------------------------------------
2       3       1               MISS
----------------------------------------
2       5       1               MISS
----------------------------------------
2       5       1               HIT
----------------------------------------
2       5       4               MISS
----------------------------------------
2       5       4               HIT
----------------------------------------
3       5       4               MISS
----------------------------------------
3       5       2               MISS
----------------------------------------
3       5       2               HIT
----------------------------------------
3       5       2               HIT

Total MISS = 7

Total HITS = 5

MISS ratio = 0.58

HIT ratio = 0.42
--------------------------------
Process exited after 122.4 seconds with return value 0
Press any key to continue . . .
```

**CONCLUSION**

Thus, LRU(Last Recently Used) algorithm helps in replacing a new page with a page that has been less frequently called so as to make efficient use of memory. Though it's a stack based page replacement algorithm, it consumes too much time in finding the page and deleting it.

| | |
|---|---|
| **REFERENCES** | 1. William Stallings, "Computer Organization and Architecture: Designing for Performance",Pearson Publication, 10 th Edition, 2013<br>2. B. Govindarajulu, "Computer Architecture and Organization: Design Principlesand Applications", Second Edition, McGraw•Hill (India) |