Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.

OS / Even Sem 2021-22 / Experiment 11.

YASH SARANG.

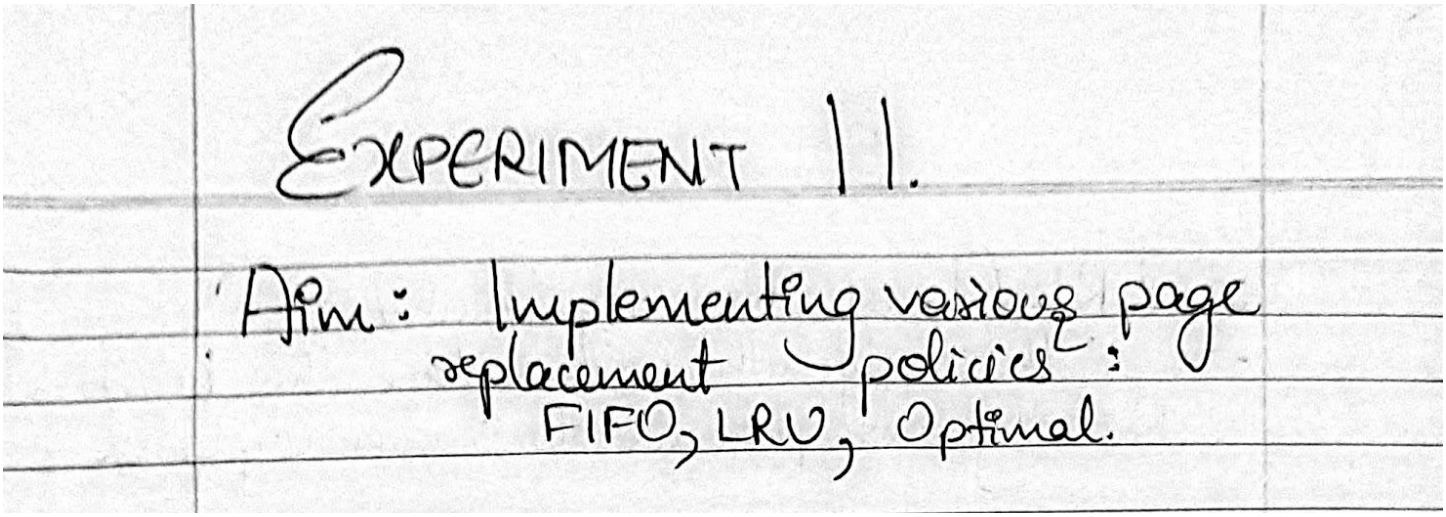
47 / D6AD.

EXPERIMENT - 11.

Page Replacement Policies.

Aim: Implementing Various page replacement policies: FIFO, Optimal, LRU.

Theory:



Theory:

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

The target of all the algorithms are to reduce page faults.

There are 3 page replacement algorithms:

1) FIFO (First in First Out)
→ Simplest page replacement algorithm. The OS tracks all the pages of memory in a queue, and the oldest page in front of the queue is selected for removal when a page needs to be replaced.

2) Optimal Page Replacement.
→ In this algorithm, the pages are replaced which would not be used for a the longest duration in the future. This policy is perfect, but not possible in practice as the OS cannot know the future requests.

3) LRU (Least Recently Used)
→ This algorith will replace the Least Recently used page as the name explains.

Code:

1. FIFO

```c
#include <stdio.h>
int main()
{
    int incomingStream[] = {4, 1, 2, 4, 5};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;
    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);
    printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
    int temp[frames];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }
    for(m = 0; m < pages; m++)
    {
        s = 0;
        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
        pageFaults++;
        if((pageFaults <= frames) && (s == 0))
        {
            temp[m] = incomingStream[m];
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }
        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }
    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```

Output:

```
Incoming Frame 1 Frame 2 Frame 3
4         4       -       -
1         4       1       -
2         4       1       2
4         4       1       2
5         5       1       2
Total Page Faults: 4
```

2. LRU

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

  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("\n");
    for (j = 0; j < no_of_frames; ++j) {
      printf("%d\t", frames[j]);
    }
  }
  printf("\n\nTotal Page Faults = %d", faults);
  return 0;
}
```

Output:

```
Enter number of frames: 3

Enter number of pages: 6

Enter reference string: 5 7 5 6 7 3

5 -1 -1

5 7 -1

5 7 -1

5 7 6

5 7 6

3 7 6

Total Page Faults = 4
```

Conclusion:-

Conclusion:

We have studied, learned and implemented different page replacement policies.

The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analysed against it.