 ***DEPARTMENT OF COMPUTER ENGINEERING***

Experiment No.

|  |  |
| --- | --- |
| Semester | S.E. Semester IV – Computer Engineering |
| Subject | Operating System |
| Subject Professor In-charge | SNA |
| Assisting Teachers | Ms. Rasika Ransing |
| Laboratory | M310B – Computer Engineering Laboratory |

|  |  |  |
| --- | --- | --- |
| Student Name | Amir Shaikh | |
| Roll Number | 18102A0053 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment Number |  | |
| Experiment Title | To implement Page Replacement Algorithms – LFU, Optimal | |
| Resources / Apparatus Required | Hardware: PC | Software:  Compiler |
| Objectives  (Skill Set / Knowledge Tested / Imparted) |  | |
| Theory: | In a computer operating system that uses paging for virtual memory management, page replacement algorithms decide which memory pages to page out, sometimes called swap out, or write to disk, when a page of memory needs to be allocated. Page replacement happens when a requested page is not in memory (page fault) and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.  When the page that was selected for replacement and paged out is referenced again it has to be paged in (read in from disk), and this involves waiting for I/O completion. This determines the quality of the page replacement algorithm: the less time waiting for page-ins, the better the algorithm. A page replacement algorithm looks at the limited information about accesses to the pages provided by hardware, and tries to guess which pages should be replaced to minimize the total number of page misses, while balancing this with the costs (primary storage and processor time) of the algorithm itself.  The page replacing problem is a typical online problem from the competitive analysis perspective in the sense that the optimal deterministic algorithm is known.  **Optimal**  In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.  **Least Frequently Used(LFU)**  In LFU Page Replacement method, the page with the minimum count is selected for replacement with the page that needs to enter into the system. | |
| Code | **LFU:**  #include<stdio.h>  int main()  {  int total\_frames, total\_pages=20, hit = 0;  int pages[20]={7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1}, frame[10], arr[20], time[20];  int m, n, page, flag, k, minimum\_time, temp;  printf("Enter Total Number of Frames:\t");  scanf("%d", &total\_frames);  for(m = 0; m < total\_frames; m++)  {  frame[m] = -1;  }  for(m = 0; m < 25; m++)  {  arr[m] = 0;  }  printf("\n");  for(m = 0; m < total\_pages; m++)  {  arr[pages[m]]++;  time[pages[m]] = m;  flag = 1;  k = frame[0];  for(n = 0; n < total\_frames; n++)  {  if(frame[n] == -1 || frame[n] == pages[m])  {  hit++;  flag = 0;  frame[n] = pages[m];  break;  }  if(arr[k] > arr[frame[n]])  {  k = frame[n];  }  }  if(flag)  {  minimum\_time = 25;  for(n = 0; n < total\_frames; n++)  {  if(arr[frame[n]] == arr[k] && time[frame[n]] < minimum\_time)  {  temp = n;  minimum\_time = time[frame[n]];  }  }  arr[frame[temp]] = 0;  frame[temp] = pages[m];  }  for(n = 0; n < total\_frames; n++)  {  printf("%d\t", frame[n]);  }  printf("\n");  }  printf("Page Hit:\t%d\n", hit);  return 0;  }  **Optimal:**  def OPT(size, pages):  SIZE = size  count = 0  memory = []  faults = 0  x = 0  for page in pages:  if memory.count(page) == 0 and count < SIZE:  memory.append(page)  count += 1  faults += 1  elif memory.count(page) == 0 and count == SIZE:  future = -1  for i in memory:  if pages[x:].count(i) == 0:  evictedPage = i  break  else:  index = pages[x:].index(i)  if index > future:  future = index  evictedPage = i  p = memory.index(evictedPage)  memory.remove(evictedPage)  memory.insert(p, page)  faults += 1  elif memory.count(page) > 0:  pass  x += 1  return faults  def main():  pages = (7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1)  print('Data to be inserted :', pages)  size = int(input("Enter the size of page: "))  print('OPT '+str(OPT(size, pages))+' page faults.')  if \_\_name\_\_ == "\_\_main\_\_":  main() | |
| Output |  | |