 ***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 | B. Niranjana | |
| Roll Number | 18104A0014 | |
| Grade and Subject Teacher’s Signature |  |  |

|  |  |  |
| --- | --- | --- |
| Experiment Number |  | |
| Experiment Title | To implement Page Replacement Algorithms – FIFO, LRU | |
| 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.  **First In First Out (FIFO)**  This is the 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.  **Least Recently Used**  In this algorithm page will be replaced which is least recently used. | |
| Code | import sys  def FIFO(size, pages):  SIZE = size  count = 0  memory = []  faults = 0  fifoIndex = 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:  memory[fifoIndex] = page  fifoIndex = (fifoIndex + 1) % SIZE  faults += 1  elif memory.count(page) > 0:  pass  return faults  def LRU(size, pages):  SIZE = size  count = 0  memory = []  faults = 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:  memory.pop(0)  memory.append(page)  faults += 1  elif memory.count(page) > 0:  memory.remove(page)  memory.append(page)  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)  size = int(input("Enter"))  print('FIFO '+str(FIFO(size, pages))+' page faults.')  print('LRU '+str(LRU(size, pages))+' page faults.')  if \_\_name\_\_ == "\_\_main\_\_":  main() | |
| Output |  | |