| | | | |
|---|---|---|---|
| **Batch:** | **B2** | **Roll No.:** | **110** |

**Experiment / assignment / tutorial No.____7___**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**TITLE :**Implementation of FIFO Page Replacement Algorithm

**AIM:** The FIFO algorithm uses the principle that the block in the set which has been in for the longest time will be replaced

---

**Expected OUTCOME of Experiment: (Mention CO/CO's attained here)**

CO3- Learn and evaluate memory organization and cache structure

---

**Books/ Journals/ Websites referred:**

**1.** Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.

**2.** William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

**3**. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization", First Edition, Wiley-India.

---

**Pre Lab/ Prior Concepts:**

T he FIFO algorithm uses the principle that the block in the set which has been in the block for the longest time is replaced. FIFO is easily implemented as a round robin or criteria buffer technique. The data structure used for implementation is a queue. Assume that the number of cache pages is three. Let the request to this cache is shown alongside.

**Algorithm:**

1. A hit is said to be occurred when a memory location requested is already in the cache.

2. When cache is not full, the number of blocks is added.

3. When cache is full, the block is replaced which was added first

**Design Steps:**

1. Start
2. Get input as memory block to be added to cache
3. Consider an element of the array
4. If cache is not full, add element to the cache array
5. If cache is full, check if element is already present
6. If it is hit is incremented
7. If not, element is added to cache removing first element (which is in first).
8. Repeat step 3 to 7 for remaining elements
9. Display the cache at very instance of step 8
10. Print hit ratio
11. End.

**Example:**

[1]

[2, 1]

[3, 2, 1]

[5, 3, 2, 1]

[5, 3, 2, 1]

[4, 5, 3, 2]

[4, 5, 3, 2]

2

SIZE = 4

memory = []

```
hit = 0



def add(a):
    global hit

    if (len(memory) != SIZE):
        memory.insert(0, a)
        print(memory)
        return
    if (a in memory):
        hit += 1
        print(memory)
        return
    else:
        memory.insert(0, a)
        memory.pop(-1)
        print(memory)



add(1)
add(2)
add(3)
add(5)
```

```
add(1)
add(4)
add(4)
print(hit)
```

**Post Lab Descriptive Questions**

**1. What is meant by memory interleaving?**

Interleaved memory is designed to compensate for the relatively slow speed of dynamic random-access memory (DRAM) or core memory by spreading memory addresses evenly across memory banks. In this way, contiguous memory reads and writes use each memory bank, resulting in higher memory throughput due to reduced waiting for memory banks to become ready for the operations.

ref- https://www.javatpoint.com/what-is-interleaved-memory

Memory interleaving is Dividing the memory into different banks for parallel accessing.

**2. Explain Paging Concept?**

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory.

ref- https://www.geeksforgeeks.org/paging-in-operating-system/

**Conclusion**

Thus we have understood FIFO algorithm in page replacement. By FIFO algorithm we replace the page which has come in first by the new required page. It is lesser efficiane than LRU in some cases. Stacks can be used to implement this algorithm.

**Date: 24 NOV 22** **Signature of faculty in-charge**