Batch: B2 Roll No.: 16010124107

Experiment / assignment / tutorial :- 7

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)

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:

The 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:

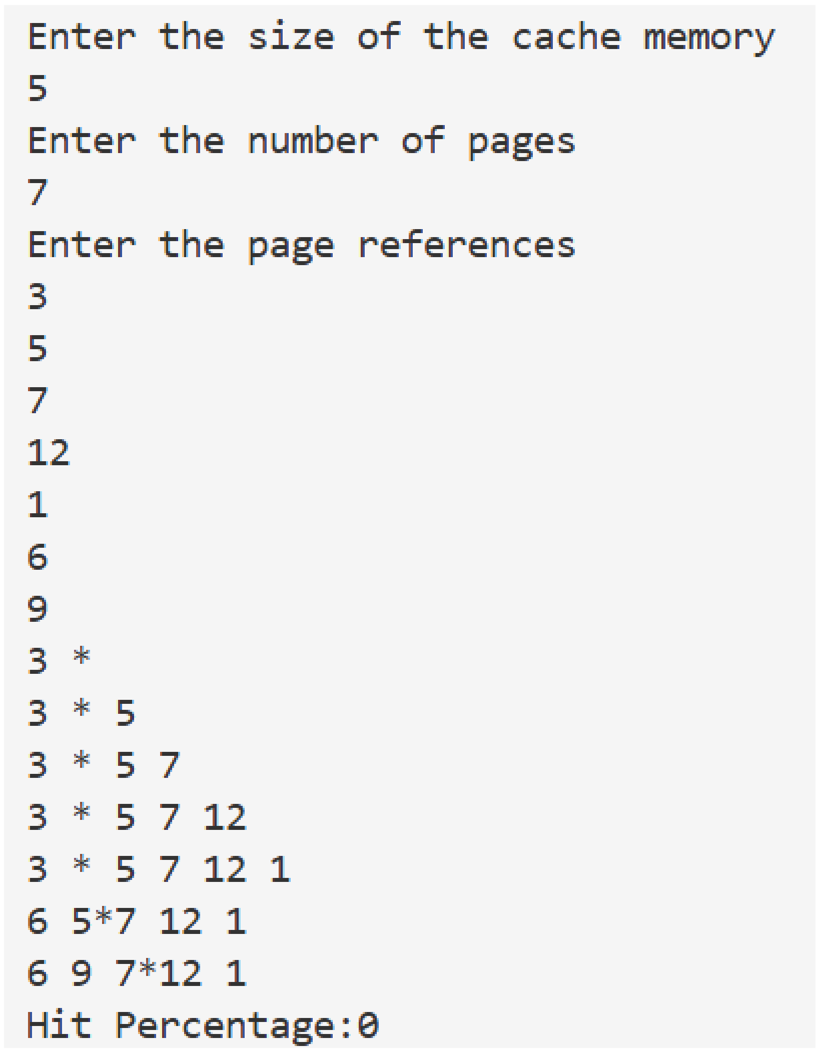
1. A hit is said to be occurred when a memory location requested is already in the cache.
2. When cache is not full, the number of blocks is added.
3. When cache is full, the block is replaced which was added first

Design Steps:
Example:

CODE:-

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cout << "Enter the size of the cache memory\n";
    int cap;
    cin >> cap;

    cout << "Enter the number of pages\n";
    int n;
    cin >> n;

    vector<int> v;
    int c = 0;
    int hit = 0;

    cout << "Enter the page references\n";
    vector<int> ref(n);
    for (auto &i : ref) {
        cin >> i;
    }

    queue<int> q;

    for (int i = 0; i < n; i++) {
        int p = ref[i];

        if (c < cap) {
            int r = q.empty() ? -1 : q.front();
            if (find(v.begin(), v.end(), p) != v.end())  {
                for (int j = 0; j < c; j++) {
                    if (v[j] != r) {
                        cout << v[j] << " ";
                    } else {
                        cout << v[j] << "*";
                    }
                }

                cout << "Hit\n";
                hit++;
                continue;
            }

            q.push(p);
            v.push_back(p);
            c++;
            r = q.empty() ? -1 : q.front();
            for (int j = 0; j < c; j++) {
                if (v[j] != r) {
                    cout << v[j] << " ";
                } else {
                    cout << v[j] << " * ";
                }
            }
        } else {
            int r = q.empty() ? -1 : q.front();
            if (find(v.begin(), v.end(), p) != v.end()) {
                for (int j = 0; j < c; j++) {
                    if (v[j] != r) {
                        cout << v[j] << " ";
                    } else {
                        cout << v[j] << "*";
                    }
                }
                cout << "Hit\n";
                hit++;
                continue;
            }
            int x = q.front();
            q.pop();

            replace(v.begin(), v.end(), x, p);
            q.push(p);
            r = q.empty() ? -1 : q.front();

            for (int j = 0; j < c; j++) {
                if (v[j] != r) {
                    cout << v[j] << " ";
                } else {
                    cout << v[j] << "*";
                }
            }
        }
        cout << "\n";
    }

    cout << "Hit Percentage:" << (float)hit / n << "\n";

    return 0;
}
```

SCREENSHOT:-



```
Enter the size of the cache memory
5
Enter the number of pages
7
Enter the page references
3
5
7
12
1
6
9
3 *
3 * 5
3 * 5 7
3 * 5 7 12
3 * 5 7 12 1
6 5*7 12 1
6 9 7*12 1
Hit Percentage:0
```

Post Lab Descriptive Questions

1. What is meant by memory interleaving?

Memory interleaving divides main memory into several independent banks to increase access speed. Instead of waiting for one memory block to respond, the CPU sends consecutive requests to different banks simultaneously. It's like multiple cashiers billing one customer's items together, reducing wait time and speeding data transfer.

2. Explain Paging Concept?

Paging is a memory management technique that divides memory into fixed-size blocks. A program's logical memory is split into pages, and physical memory into equal frames. Pages can be stored in any free frame, even non-contiguously. A page table tracks each page's location in RAM.

Conclusion:-

In conclusion, paging efficiently manages memory by allowing non-contiguous storage, reducing fragmentation, and simplifying memory allocation through the use of a page table.

Date: 02/11/2025