# Project Report

Operating Systems

CSN-232

Coding Assignment - 2 LRU Implementations

## Group members


| | |
|---|---|
| Ashutosh Chaubey | 17114015 |
| Ayush Agarwal | 17114017 |
| Bhanu Pratap | 17114019 |
| Shashank Kashyap | 17114070 |
| Piyush Makwana | 17114056 |
| Prateek Mali | 17114059 |
| Ritik Kumar | 17114063 |
| Mula Ram | 17114049 |

# Problem Statement

Implement the following algorithms in C or C++ without using existing/predefined classes.

i. LRU - Counter Method

ii. LRU - Stack Method

iii. LRU - Aging Register Method

iv. Approx. LRU - Clock Method

**Page replacement:** In any page table, the size of the table plays a major role. As the size of the table increases, page query naturally becomes slower. As such, in a size constrained page table, we need to update the table removing the older pages to accommodate the newer ones. The choice of the pages to be removed plays a vital role in determining the total page misses and thus affecting the overall performance of the device.
A lot of algorithms have been developed to decide the page to be replaced. Common ones of them being,

- Least recently used (LRU) page replacement algorithm
- The optimal Page Replacement Algorithm
- First In First Out (FIFO)

**LRU:** The Least Recently Used (LRU) Page Replacement Algorithm is a good Greedy algorithm. It is based on the locality of reference. Pages which are recently used will have more chance to be used again. Thus if a page is to be removed from the page table, it should be the least recently accessed one.

LRU is often used as a page replacement algorithm and is often considered good.
The 4 required LRU are:

- **Counter Method:** In this method, every page table entry stores an extra field about the time of use of them. A counter counts every memory reference made. When a new page table entry is added or updated, the corresponding field is updated with the current value of the counter. At the time of page replacement, the one with the least count shall be removed.
- **Stack Method:** In this method, the page table maintains a stack of page numbers. Whenever a page is referenced, it is removed from the stack and put on the top. Thus

the most referenced page is always in the top and the least referenced page slides down to the bottom of the stack. At the time of page replacement, the one at the bottom of the stack shall be removed.

- **Aging Register Method:** In this method, every page table entry stores an extra reference counter field, instead of a simple counter. After every page table reference, the reference bits are shifted right( divided by 2) and a new reference bit is added. For instance, if a page has referenced bits 1,0,0,1,1,0 in the past 6 clock ticks, its referenced counter will look like this: 10000000, 01000000, 00100000, 10010000, 11001000, 01100100. At the time of page replacement, the one with the least reference counter shall be removed.
- **Clock Method:** In this method, every page table entry stores an additional referenced R bit which is used to decide if the page should be removed. The algorithm keeps a circular list of pages in memory, with the "hand" (iterator) pointing to the last examined page frame in the list. When a page fault occurs and no empty frames exist, then the R bit is inspected at the hand's location. If R is 0, the new page is put in place of the page the "hand" points to, and the hand is advanced one position. Otherwise, the R bit is cleared, then the clock hand is incremented and the process is repeated until a page is replaced.

---

## Corner cases testing of the algorithms

The common corner case for LRU algorithms for a given page table size (say 5 entries) are:

1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 ....

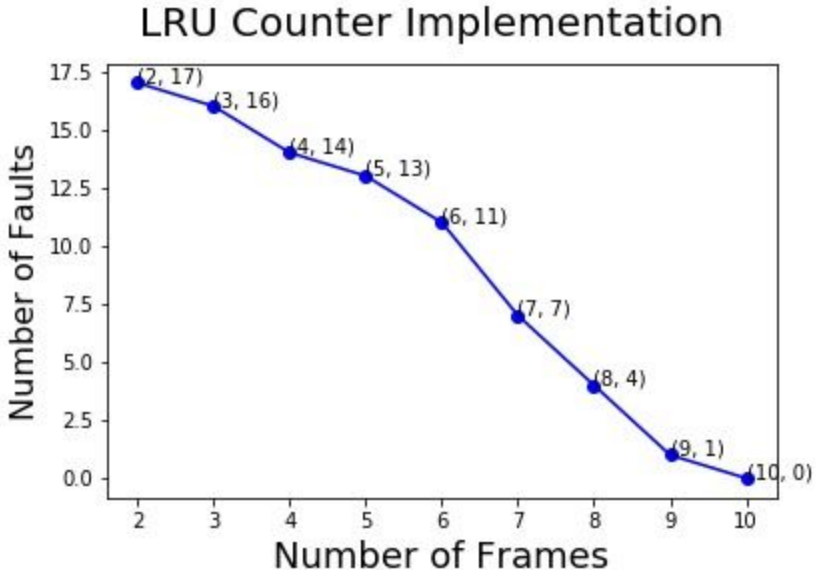
I.e. if the number of frames is n, then the corner case is a sequence of requests from 1 to n+1 done repeatedly.

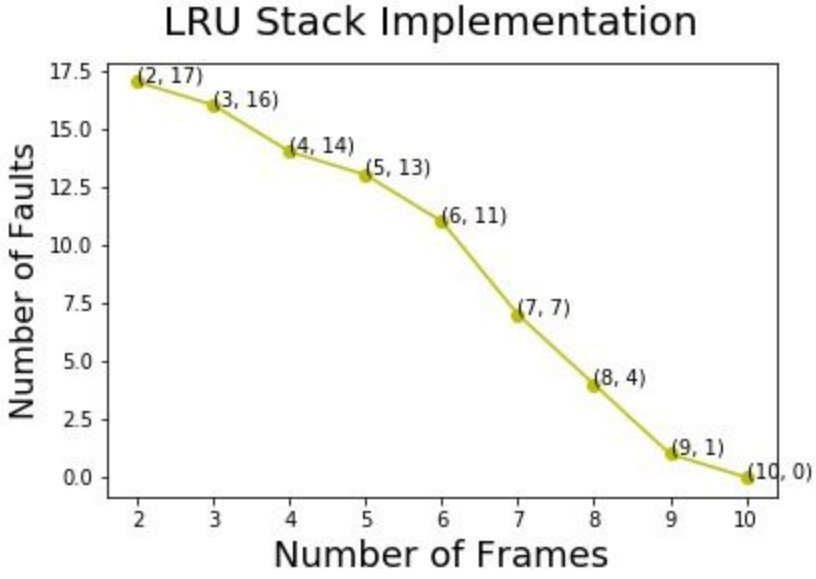
On requesting the pages according to the sequence above, few conclusions were drawn. The page faults were exactly the same in all but clock method. The clock method being approximate LRU performed slightly differently among them. Coming to the time performance, Aging approach performed the worst while clock and the counter were best among all. Some of the relevant graphs have been added in the sections below for further clarity.

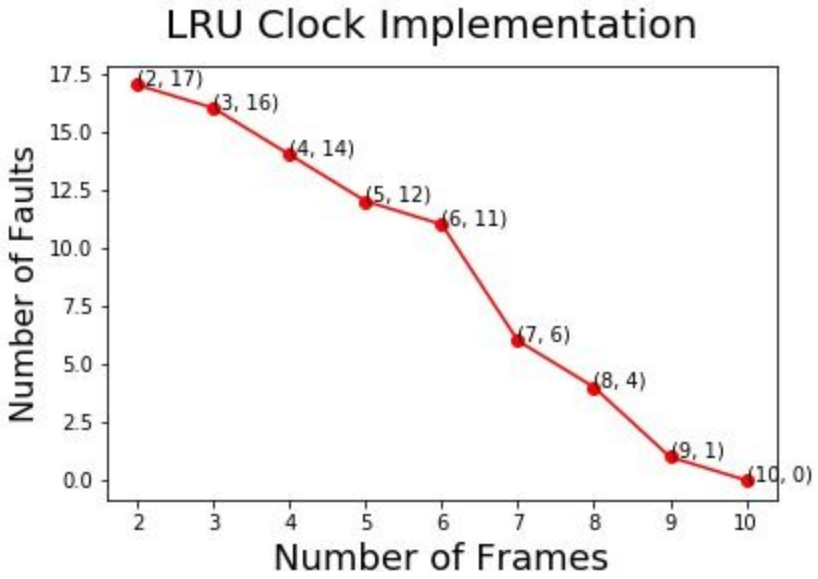
---

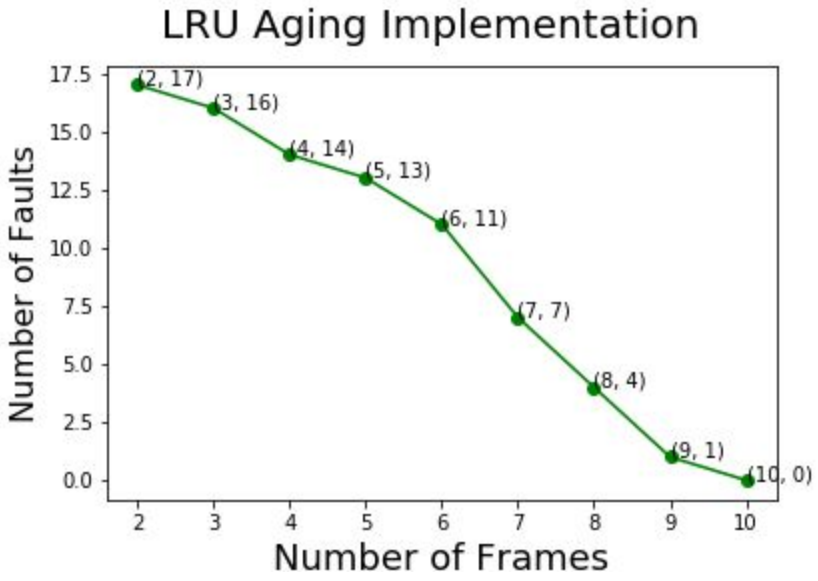
# The graph for the number of frames vs. the number of page faults.

The graphs of the number of page faults vs. number of frames have been added below:





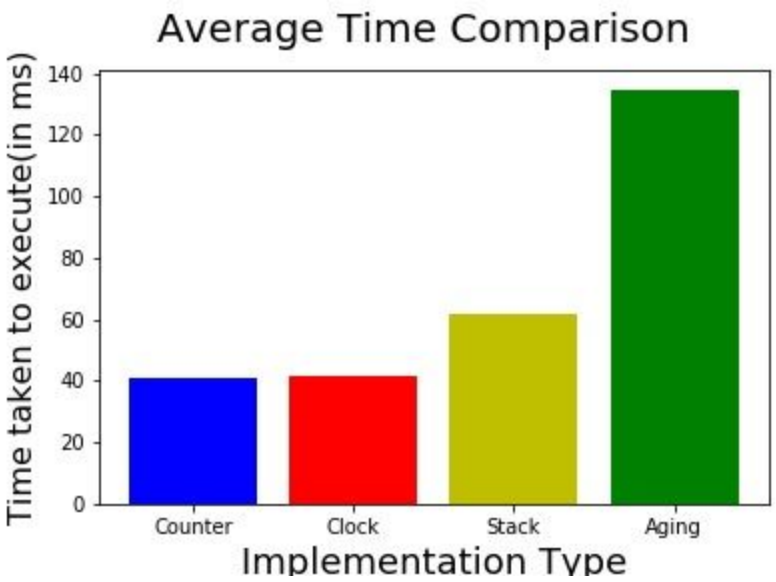




An analysis of the graph is added in the last segment of the report.

---

# The graph for the average time taken to execute the algorithms for different corner cases.

The graph for the average time taken to execute the algorithms for different corner cases has been added below:




An analysis of the graph is added in the last segment of the report.

---

# Complexity analysis.

The algorithms vary significantly in terms of space and time complexity.

Taking about **space complexity**, the counter method stores the counter data per entry. Thus an appropriate space (say a byte) should be allocated to each and every entry. In the case of the stack method, a separate stack is required which would contain pointers to every entry, with some other information for identifying the page entry. In the clock method, a single bit is enough

per entry to facilitate the working of the algorithm. Whereas in Aging, a byte or so of the counter is enough for the algorithm to work, per entry.

**Time Complexity:**
For Counter Algorithm: Time complexity to determine minimum counter value is O(n) where n is the number of frames and O(1) is the time complexity for replacing an an appropriate page in the table.
For Stack Algorithm: The time complexity for table lookup is O(n) where n is the number of frames and O(1) is the time complexity for replacing an appropriate page in the table.
For Aging Algorithm: Time complexity of this lookup is O(n) where n is the number of frames in the system. Time complexity for page replacement is also O(n). In the graph the time taken is almost double in comparison to other implementations but it's still in the order of n. The extra time is because we have to update the registers of all the pages instead of just one during lookup.
For Clock Algorithm: Time complexity for table lookup is O(n) where n is the number of frames and the time complexity of replacement is also O(n).

---

# Critical analysis of the results:

We have implemented basic algorithms to implement LRU page replacement algorithms. The graphs attached above are the testimony for the implementations.

From the number of faults vs the number of frames graph, it is clear that as the number of frames increase, the page faults generally decrease to 0. The 3 algorithms (Counter, Stack and the Aging) being the implementation of LRU has the exact same graph. The clock method is just an approximate LRU and hence its graph is slightly different from the others.

The first 3 differs among each other in terms of their complexities. This is evident from the graph above. Here the counter takes the least time with the clock method whereas the stack takes intermediate time. It is the Aging method which is the most time consuming, given the conditions are the same. Hence, it can be easily concluded that Aging method is the worst method among them in terms of time complexity.

---