“Least Recently Used (LRU) Cache Implementation.”

Sachit Sharma , Gautam Kumar and Himanshu Sharma

2021a1r027@mietjammu.in

2021a1r005@mietjammu.in

2021a1r012@mietjammu.in

Model Institute of Engineering and Technology, CSE department, Kot Bhalwal,Jammu,

Jammu and Kashmir 181122

Abstract

Least Recently Used (LRU) algorithm is a page replacement technique used for memory management. According to this method, the page which is least recently used is replaced. Therefore, in memory, any page that has been unused for a longer period of time than the others is replaced.

The process takes less time for handling instructions then would be speedy and efficient. The speed of the process is not only depends on architectural features and operational frequency, but also depends on the algorithm and data structure, which is used for that process. There are many page replacement algorithms such as least recently used (LRU), first-in-first-out (FIFO), Optimal and the combination of LRU and least frequently used (LFU) are available in memory management. We can improve the performance of page replacement algorithms by either developing a new algorithm from scratch or using augmentation of data structure and algorithm design. In this paper we use augmented doubly circular link list, skip list, splay tree and hash table and apply all these on LRU page replacement algorithm. We implement all these augmented data structures with LRU page replacement algorithm and traced using Standard Performance Evaluation

Corporation (SPEC) benchmark files. This paper shows that augmentation of data structure can improve the performance of LRU page replacement algorithm.

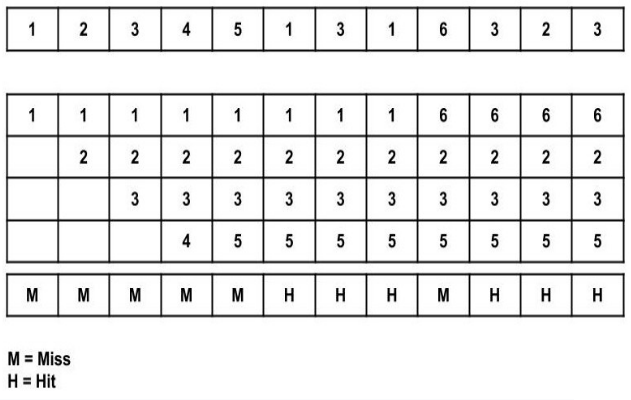
Introduction

LRU stands for Least Recently Used. As the name suggests, this algorithm is based on the strategy that whenever a page fault occurs, the least recently used page will be replaced with a new page. So, the page not utilized for the longest time in the memory (compared to all other pages) gets replaced. This strategy is known as *LRU paging*.

A *page fault* occurs when a running program tries to access a piece (or page) of memory that is not already present in the main memory (RAM). On the other hand, if that page is already present in the memory, it is called a *page hit*. The LRU page replacement algorithm comes into the picture whenever a page fault occurs.

SPECIFICATIONS:

Least Recently Used (LRU) algorithm is a page replacement technique used for memory management. According to this method, the page which is least recently used is replaced. Therefore, in memory, any page that has been unused for a longer period of time than the others is replaced.



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

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

M	M	M	M	M	H	H	H	M	H	H	H

M = Miss
H = Hit

Implementation Overview

IMPLEMENTATION:

Given a reference string: `0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4` and given that there are a total of 3 memory locations available.

1. **The first three pages, `0, 1, 2`, will give rise to 3-page faults (look at the image above).**
2. **Now that the table is filled and the next page, `3`, is not present in the table, a page must be replaced. Since `0` is the least recently used page, page `3` will replace `0`. This will result in a `1` page fault.**
3. **This process is repeated for the entire reference string.**
4. **When a number from the reference string already**

exists in the table, there is no page fault and no effort is required to replace the page. In that case, the reference page will become the recently used page (like when `0` is referenced when the page frames contain `4`, `0` and `1`.)

5. Make sure you look at whether or not a given page is recently used from the reference string.

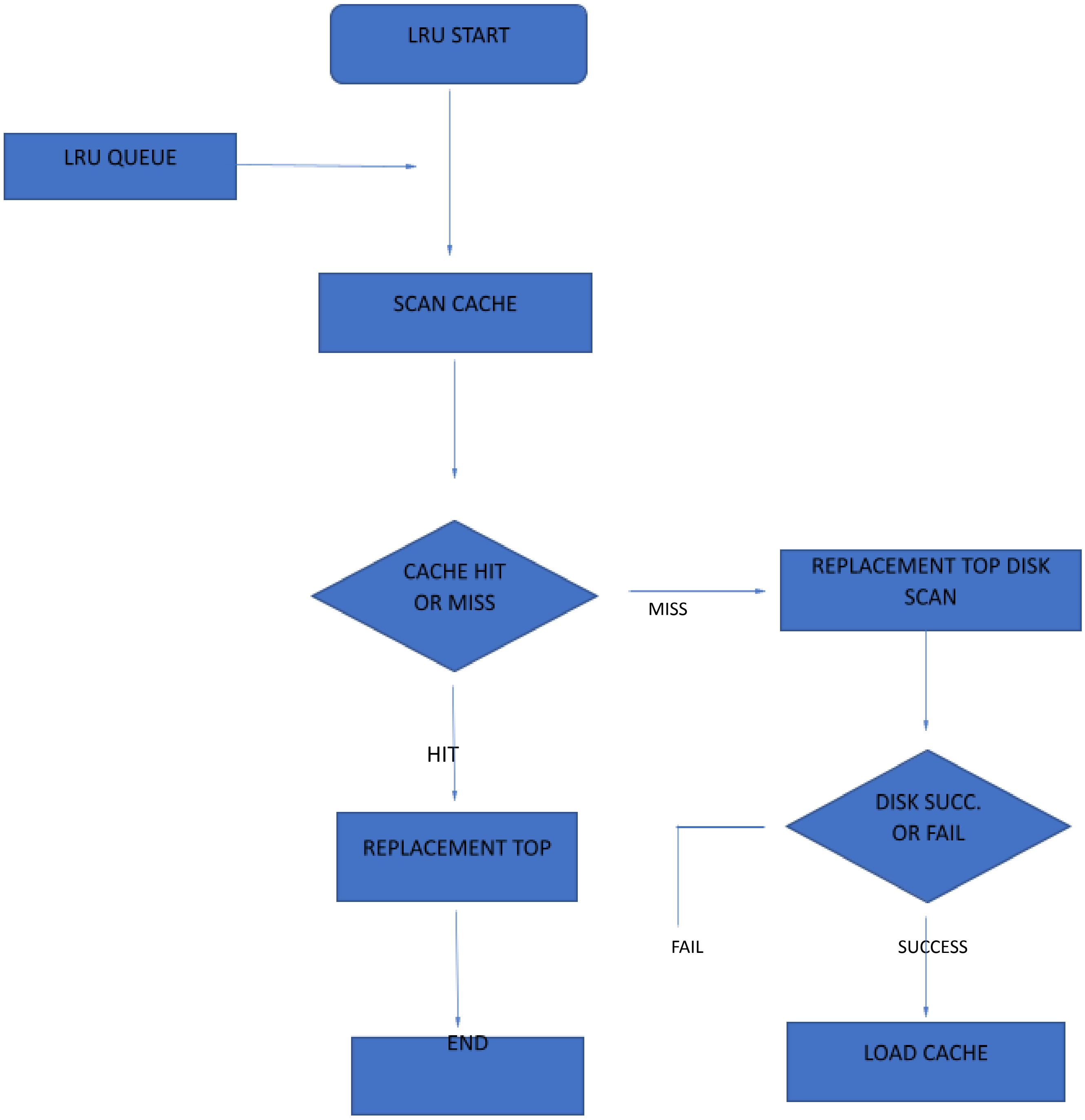
Pseudocode of LRU Algorithm:

Let us say that s is the main memory's capacity to hold pages and pages is the list containing all pages currently present in the main memory.

```
1. Iterate through the referenced pages.
    ○ If the current page is already present in pages:
        1. Remove the current page from pages.
        2. Append the current page to the end of pages.
        3. Increment page hits.
    ○ Else:
        1. Increment page faults.
        2. If pages contains less pages than its capacity s:
            ■ Append the current page into pages.
        3. Else:
            ■ Remove the first page from pages.
            ■ Append the current page at the end of pages.
2. Return the number of page hits and page faults.
```

FLOWCHART:

Flow chart



Advantages of LRU Page Replacement Algorithm:

Following are the advantages of the LRU page replacement algorithm:

- The page that has not been used for the longest time gets replaced.
- It gives fewer page faults than any other algorithm.
- The algorithm never suffers from the Belady's anomaly .
- The algorithm is capable of complete analysis.

Disadvantages of LRU Page Replacement Algorithm:

Following are the disadvantages of the LRU page replacement algorithm:

- The execution of this algorithm is complicated.
- This algorithm's execution may require significant assistance from the hardware.

Conclusion

- The LRU page replacement algorithm replaces the least recently used page in the memory with a new page.
- The LRU page replacement algorithm is based on the assumption that among all pages in the memory, the least recently used page will not be used for a long time.
- A page fault occurs when a program tries to access a memory page that is not already present in the memory.
- A page hit occurs when a program tries to access a memory page that is present in the memory.
- LRU page replacement algorithm gives the least page faults compared to other page replacement algorithms.
- The LRU algorithm requires assistance from the hardware to execute.

REFERENCES:

Code –

- https://www.geeksforgeeks.org/ipc-shared-memory/
- https://www.javatpoint.com/ipc-through-shared-memory
- https://www.tutorialspoint.com/ipc-through-shared-memory

Algorithm –

- **geeksforgeeks**
- **,Java point**
- **Edukera**
- **Wikipedia.**