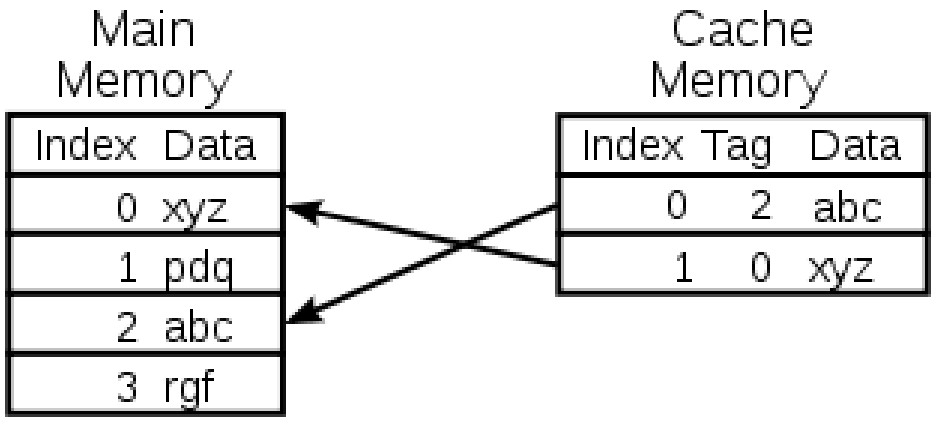
COMPUTER ORGANISATION (CEN-402)
ASSIGNMENT-2
ANZAL HUSAIN ABIDI 20BCS009

Cache Replacement Algorithms: How To Efficiently Manage The Cache Storage :

Caching ,an overview :

Caching is the process of storing some data near where It's supposed to be used rather than accessing them from an expensive origin, every time a request comes in.

Caches are everywhere. From your CPU to your browser. So there's no doubt that caching is extremely useful. implementing a high-performance cache system comes with its own set of challenges. In this post, we'll focus on cache replacement algorithms.



Cache Replacement Algorithms :

We talked about what caching is and how we can utilize it but there's a dinosaur in the room; Our cache storage is finite. Especially in caching environments where high-performance and expensive storage is used. So in short, we have no choice but to evict some objects and keep others.

Cache replacement algorithms do just that. They decide which objects can stay and which objects should be evicted.

After reviewing some of the most important algorithms we go through some of the challenges that we might encounter.

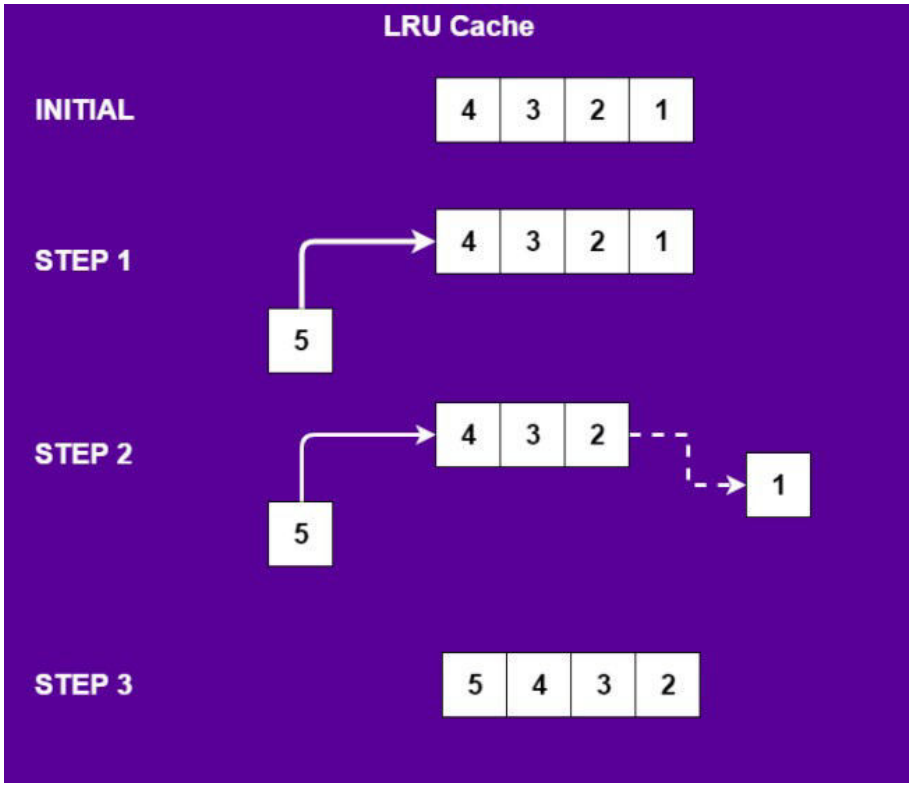
1)LRU

The least recently used (LRU) algorithm is one of the most famous cache replacement algorithms and for good reason!

As the name suggests, LRU keeps the least recently used objects at the top and evicts objects that haven't been used in a while if the list reaches the maximum capacity.

So it's simply an ordered list where objects are moved to the top every time they're accessed; pushing other objects down.

LRU is simple and providers a nice cache-hit rate for lots of use-cases.



2)LFU

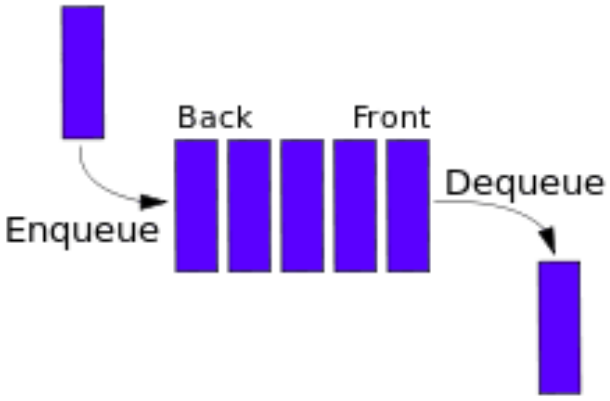
the least frequently used (LFU) algorithm works similarly to LRU except it keeps track of how many times an object was accessed instead of how recently it was accessed.

Each object has a counter that counts how many times it was accessed. When the list reaches the maximum capacity, objects with the lowest counters are evicted.

LFU has a famous problem. Imagine an object was repeatedly accessed for a short period only. Its counter increases by a magnitude compared to others so it's very hard to evict this object even if it's not accessed for a long time.

3)FIFO

FIFO (first-in-first-out) is also used as a cache replacement algorithm and behaves exactly as you would expect. Objects are added to the queue and are evicted with the same order. Even though it provides a simple and low-cost method to manage the cache but even the most used objects are eventually evicted when they're old enough.



4)Random Replacement (RR)

This algorithm randomly selects an object when it reaches maximum capacity. It has the benefit of not keeping any reference or history of objects and being very simple to implement at the same time.

This algorithm has been used in ARM processors and the famous Intel i860.