Assignment 3

24MCAT103 - Digital Fundamentals and Computer Architecture
&
24MCAT107 - Advanced Software Engineering

Subject: Simulating Cache Memory Mapping and Replacement Algorithms using Git Collaboration

Submitted By,
Jaganath Syam
Regular MCA
Roll No: 34
Submitted on,
15 December 2025

Least Recently Used

The Least Recently Used (LRU) cache replacement algorithm is a widely used strategy in memory management that replaces the data item which has not been accessed for the longest period of time when the cache is full. The core idea behind LRU is based on the principle of temporal locality, which states that data accessed recently is more likely to be accessed again in the near future. When a cache miss occurs and there is no free space, the algorithm identifies the least recently used item and removes it to make room for the new data.

LRU can be implemented using data structures such as a hash map combined with a doubly linked list, or simpler structures like lists for small-scale systems. The hash map allows fast access to cache items, while the linked list maintains the order of usage, making it easy to identify and remove the least recently used element. Due to its effectiveness in reducing cache misses, LRU is commonly used in operating systems, databases, and web caching systems, although it may have slightly higher overhead compared to simpler algorithms like FIFO.

Algorithm

Step 1: START

Step 2: Read the number of cache sets

Step 3: Read the number of ways (cache lines) per set

Step 4: Read the sequence of memory block references

Step 5: Initialize the cache by creating the required number of sets, each containing the specified number of ways

Step 6: Initialize hit counter = 0 and miss counter = 0

Step 7: For each memory block in the reference sequence, perform the following steps:

Step 7.1 : Compute the set index using:

set_index = block_number mod number_of_sets

Step 7.2 : Select the corresponding cache set.

Step 7.3 : If the block is present in the selected cache set, then:

a. Increment the hit counter

b. Update the block as most recently used

Step 7.4 : Else (block not present):

a. Increment the miss counter

b. If the cache set has free space, insert the block

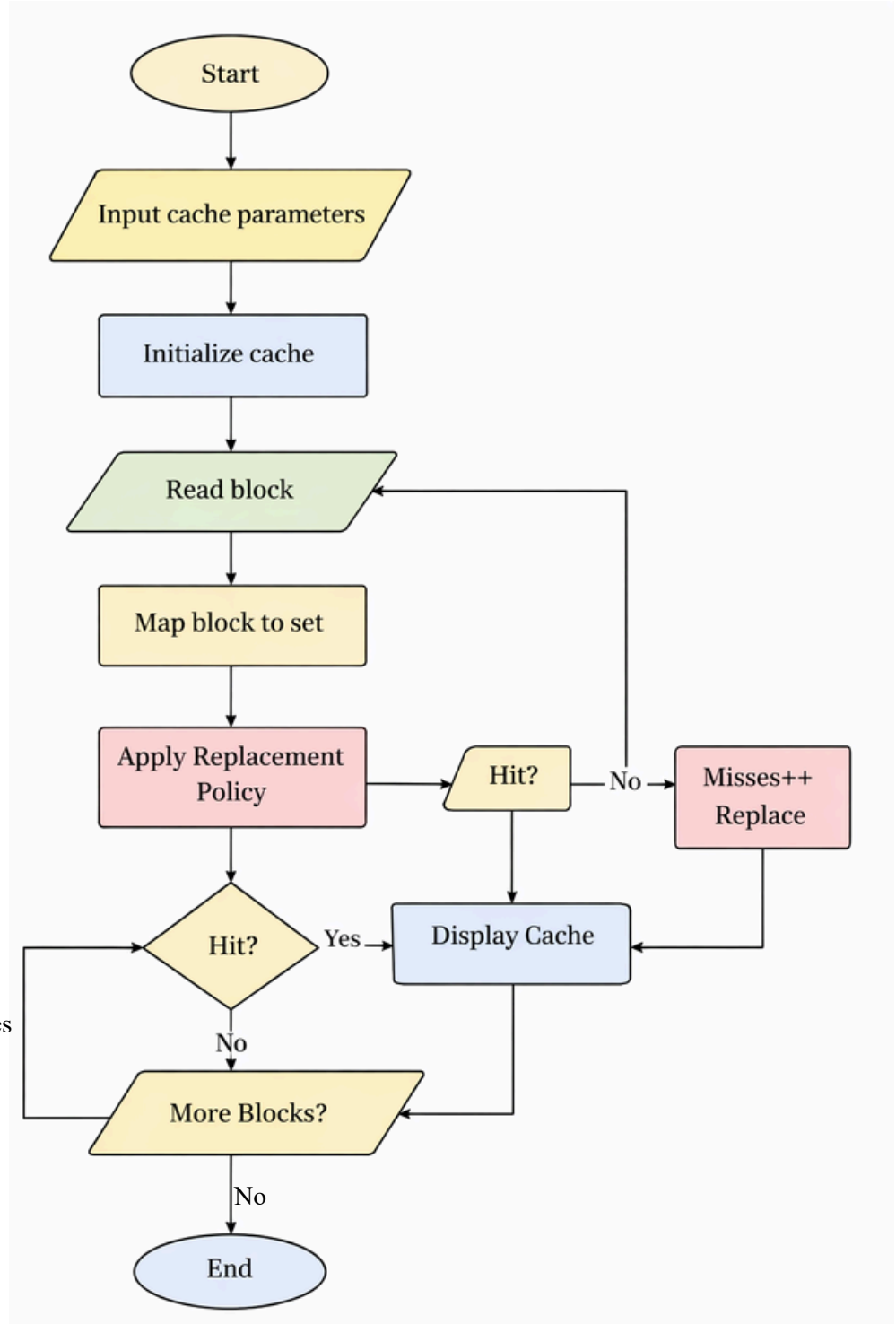
c. Else, remove the least recently used block from the set

d. Insert the new block as most recently used

Step 8: Display the contents of all cache sets after each memory reference.

Step 9: STOP

Flow Diagram



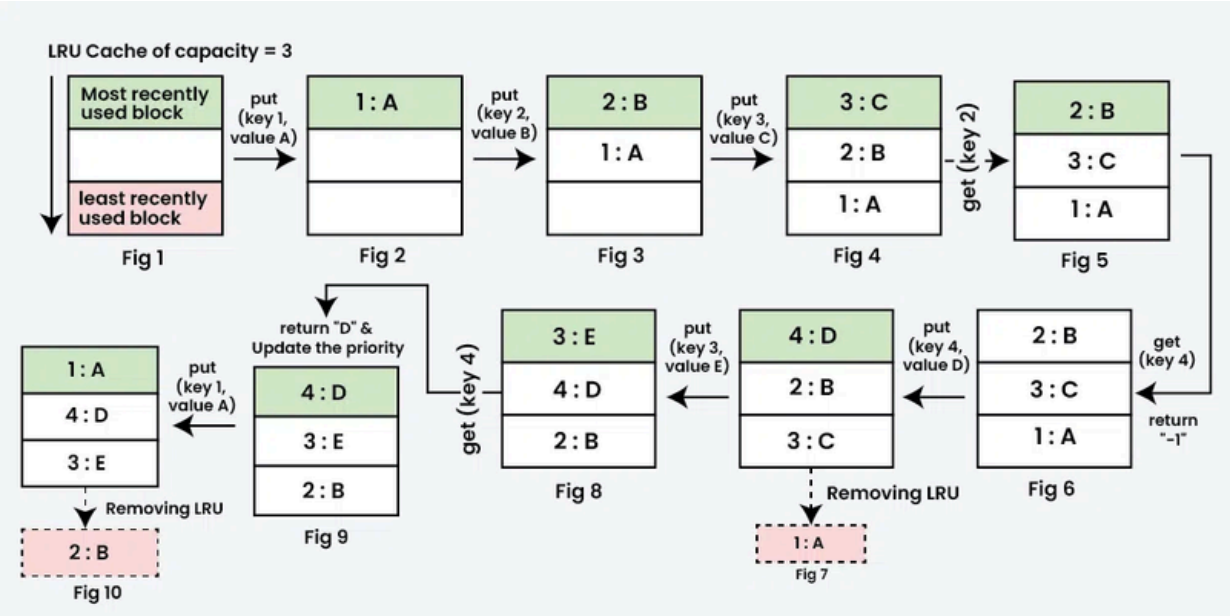
Python Program:

```
cache_size = int(input("Enter cache size: "))
cache = []
hits = 0
misses = 0

refs = input("Enter memory block references (space separated):
").split()

for block in refs:
    if block in cache:
        hits += 1
        cache.remove(block)
        cache.append(block)   # move to most recently used
        print(f"Hit  -> Cache: {cache}")
    else:
        misses += 1
        if len(cache) == cache_size:
            cache.pop(0)      # remove least recently used
        cache.append(block)
        print(f"Miss -> Cache: {cache}")

print("\nTotal Hits:", hits)
print("Total Misses:", misses)
print("Hit Ratio:", hits / (hits + misses))
```

Output

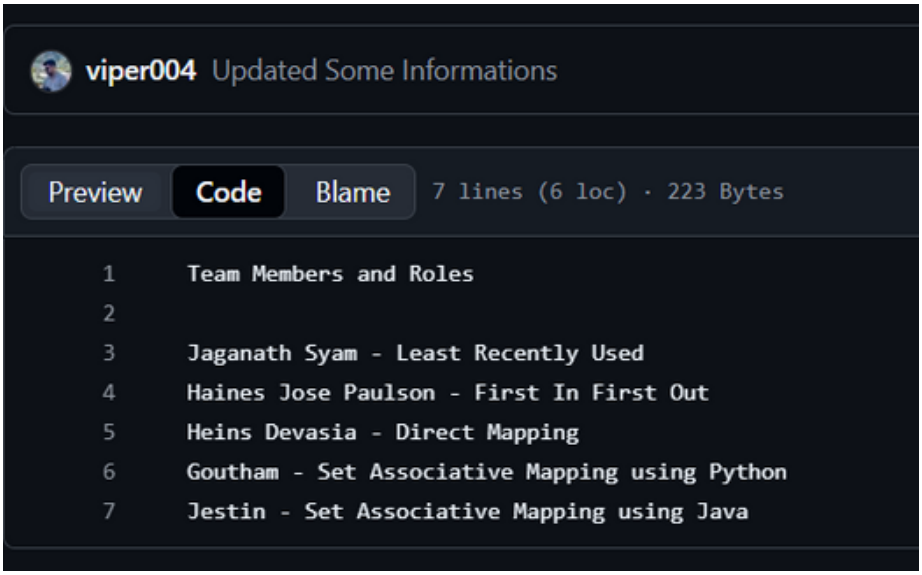
```
PS C:\Users\jagan\Desktop\df> & C:/Users/jagan/AppData/Local/Pr
Enter cache size: 3
Enter memory block references (space separated): a b c d e
Miss -> Cache: ['a']
Miss -> Cache: ['a', 'b']
Miss -> Cache: ['a', 'b', 'c']
Miss -> Cache: ['b', 'c', 'd']
Miss -> Cache: ['c', 'd', 'e']

Total Hits: 0
Total Misses: 5
Hit Ratio: 0.0
PS C:\Users\jagan\Desktop\df>
```

WorkFlow



Git Repository: [Click Here]

readme.md



git log

```
commit df96fc2013d3595eaf600450a536b9d94f2ed792 (HEAD -> lru-jagan, origin/lru-jagan)
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:41:50 2025 +0530

    .

commit f94395f8d09fe17bb4be3f80b13d0c7da2f8a89e
Merge: 2e144fa 78a6dd8
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:37:28 2025 +0530

    Merge branch 'lru-jagan' of https://github.com/viper004/df_assignment into lru-jagan

commit 2e144fa7be06898bf854cacf8a2bd2cf6cb7dca0
Author: viper004 <jagannathsyam2004@gmail.com>
commit df96fc2013d3595eaf600450a536b9d94f2ed792 (HEAD -> lru-jagan, origin/lru-jagan)
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:41:50 2025 +0530

    .

commit f94395f8d09fe17bb4be3f80b13d0c7da2f8a89e
Merge: 2e144fa 78a6dd8
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:37:28 2025 +0530

    Merge branch 'lru-jagan' of https://github.com/viper004/df_assignment into lru-jagan

commit 2e144fa7be06898bf854cacf8a2bd2cf6cb7dca0
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:33:23 2025 +0530

    .

commit 78a6dd801abfb85a60dfd61b70d123d45e305e15
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Sun Dec 14 23:29:57 2025 +0530

    added all files into a folder

commit 351b319fdea583541b1e6119a133e470c4381e51
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Sun Dec 14 23:16:31 2025 +0530

    testing a new branch creation

commit b38fd3a2dd9e0b9144639aa27dc51d0d6a2f00cd
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Wed Dec 10 23:38:48 2025 +0530

    .

commit b0afc7d61b8f0022014f0ce4efa1206d13b02ff0
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Wed Dec 10 23:32:32 2025 +0530

    .

commit 2288856835ae347a81e304723a85e6ccd5ccf34d
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Thu Dec 4 23:52:26 2025 +0530

    Reduced LOC
commit 9df9b42ecde24749886890c0fcb900d63954b2e4
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Thu Dec 4 23:48:40 2025 +0530

commit df96fc2013d3595eaf600450a536b9d94f2ed792 (HEAD -> lru-jagan, origin/lru-jagan)
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:41:50 2025 +0530

    .

commit f94395f8d09fe17bb4be3f80b13d0c7da2f8a89e
Merge: 2e144fa 78a6dd8
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:37:28 2025 +0530

    Merge branch 'lru-jagan' of https://github.com/viper004/df_assignment into lru-jagan

commit 2e144fa7be06898bf854cacf8a2bd2cf6cb7dca0
Author: viper004 <jagannathsyam2004@gmail.com>
Date:   Mon Dec 15 21:33:23 2025 +0530

    .
```

git branch

```
PS C:\Users\jagan\Desktop\df> git branch
* lru-jagan
PS C:\Users\jagan\Desktop\df>
```

Reflection

In this project, cache memory concepts were implemented using software logic by simulating the behavior of the Least Recently Used (LRU) cache replacement algorithm in Python. Since cache memory has limited capacity, the program was designed to track data usage and remove the least recently accessed element whenever the cache becomes full. This was achieved by maintaining the order of access using basic data structures such as lists or dictionaries. Whenever an element was accessed, it was moved to the most recent position, effectively representing real-world cache behavior through logical operations rather than hardware.

Git played a crucial role in collaborative coding and version management throughout the project. A shared Git repository was created, and each team member worked on an individual branch. This branch-based workflow allowed multiple contributors to develop and test features independently without interfering with others' work. Git also provided a clear history of changes through commits, making it easier to track progress, identify issues, and revert to previous versions if necessary. Merging branches helped integrate individual contributions in an organized manner.

Several challenges were faced during the project. Understanding how to accurately represent cache access patterns using software logic required careful planning. Additionally, resolving merge conflicts and maintaining consistent code structure across branches was initially difficult. However, these challenges provided valuable learning experiences. The project improved my understanding of cache replacement algorithms, strengthened my Python programming skills, and enhanced my knowledge of version control systems. Overall, this experience highlighted the importance of efficient algorithms and proper collaboration tools in software development.