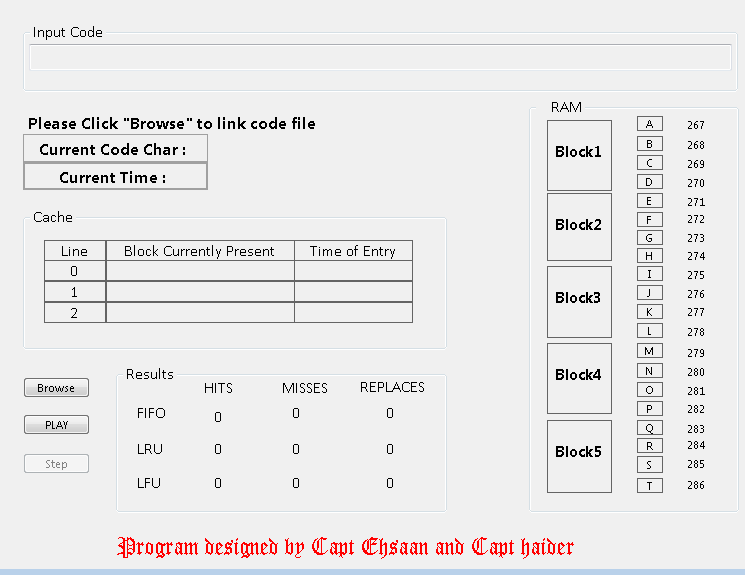
**CACHE ASSOCIATIVE MAPPING SIMULATOR**

**TABLE OF CONTENTS**

INTRODUCTION…………………………………………………………………………………3

PROBLEM………………………………………………………………………………………...4

DESIGN……………………………………………………………………………………………4

WORKING…………………………………………………………………………………………6

CONCEPTS………………………………………………………………………………………..7

CLASS DIAGRAM…………………………………………………………………………………7

SEQUENCE DIAGRAM…………………………………………………………………………..8

REFERENCES & CREDITS……………………………………………………………………..9

# INTRODUCTION

Cache is a small memory storage device embedded in the processor to facilitate the quick retrieval of data needed to be frequently used by the processor. Whenever the processor asks for the data, it searches in the cache and if available fetches it directly from there or else brings it from the main memory.

The retrieval of data from the cache is called **hit** and it is faster than the retrieving data directly from the main memory.

If processor is unable to find the data in cache, it is called as cache **miss.**

Such patterns which instruct the cache to fetch the data from Ram which is required by the processor are called mappings. Generally, there are three types of Cache Mappings.

* Direct Mapping
* Associative Mapping

It uses a variety of Replacement Algorithms for the placement of blocks from the Memory on Cache Lines. These Algorithms are of following three types:

* + FIFO (First in First Out): The memory block first placed or the oldest memory block placed in the cache is first replaced by the new block. This replacement is done on the basis of the entry time of the block into the cache.
  + LRU (Least Recently Used): The block that is least recently used by the processor is replaced with the new block. This is done on the criteria of time of last usage.
  + LFU (Least Frequently Used): The block that is least used by the processor is replaced depending on the frequency of its usage.
* Set Associative

# AIM

To illustrate the trace of associative mapping for a code that shows the sequence of bytes accessed in RAM. The illustration must show the Byte, the block and the movement into block.

# SOLUTION

The aim was served by creating an application known as the “Associative Mapping Cache Simulator”. This application was built to show the working of the cache with the help of Graphical User Interface.To serve the purpose, this application was built on C sharp platform using Windows Form Applications.

# DESIGN

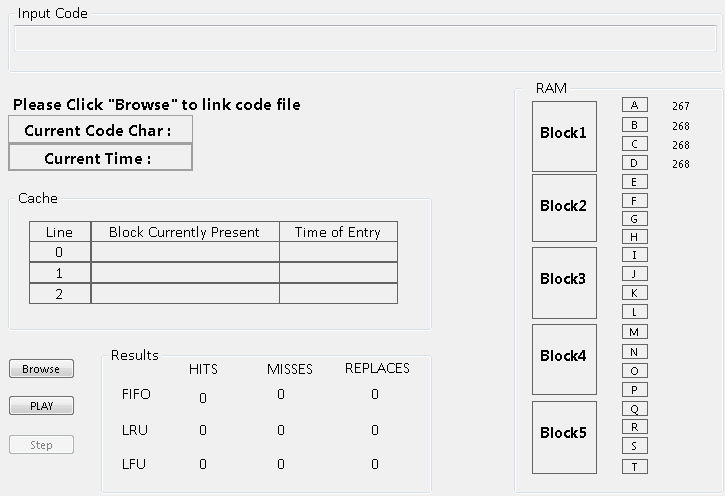
The program is composed of a single Form. The user is given with option to select a file and then run simulation.



# RAM & Cache Design:

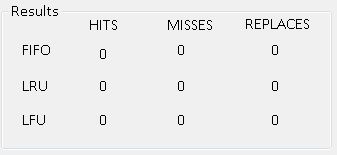
To implement the RAM and the Cache, text boxes and buttons were used. These were customized to meet the graphical requirements of the program.

The text boxes are so modified as such to display each RAM location as a separate character. The Cache line is also made of textboxes which displays the current blocks present in the cache. The current code char displays the code that is being executed currently and current times shows the time of code execution whereas the next code char shows the upcoming code character.



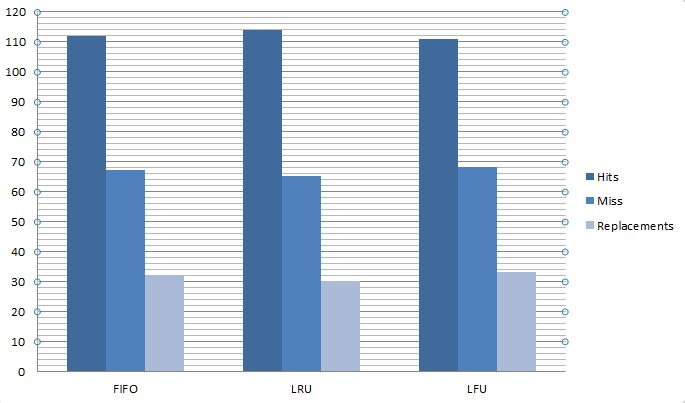
# STATISTICS

A table displays the statistics of the results of all three replacement algorithms. It shows the hits, misses and replacements separately for all algorithms. A count is maintained for this purpose and is displayed.

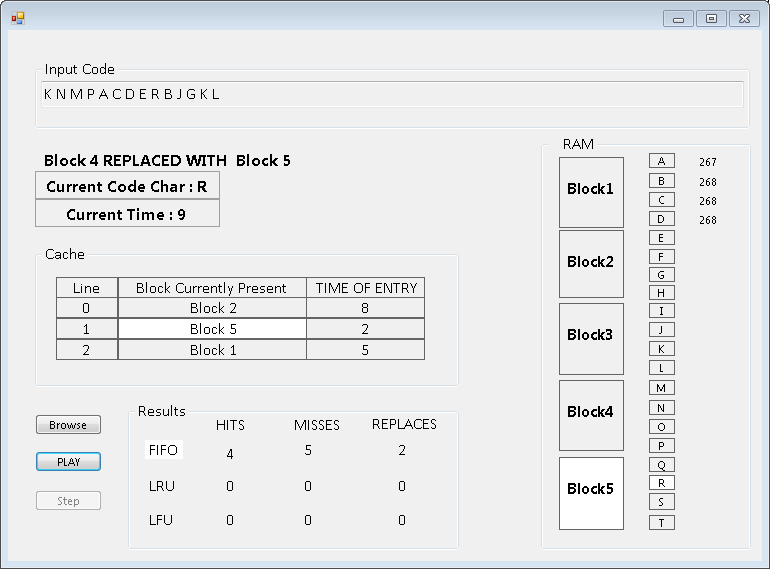


# GRAPHICAL ANALYSIS

Furthermore, the graph was also plotted to analyze the results of 15different codes. The conclusion was almost same for all the three algorithms. Thus it can be said that the efficiency of the algorithms highly depend on the code being executed.



# WORKING OF THE PROGRAM



The working of the program goes as follows.

1. Open code file:

The program first browses for the code file in .txt format. With the help of exception handling, a check is maintained for which the program does not accept any other file format except the notepad (\*.txt) files.

1. Start Simulation:

Once the code file is browsed, the program now provides the option to either start simulation all at once or step by step. The flow goes as first FIFO is implemented for the code and followed by LRU and LFU. The hits, misses and replacements for every algorithm is maintained in a table.

# CONCEPTS

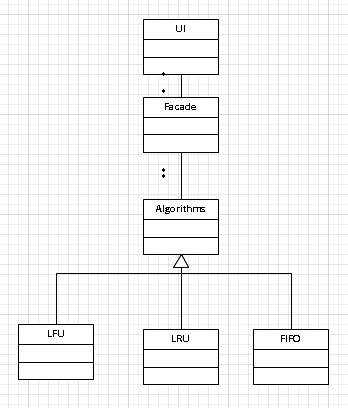
The following concepts are implemented during the construction of the program.

1. Exception Handling
2. File Handling
3. System Threading
4. System Sleep

As such no classes for the algorithms are maintained.

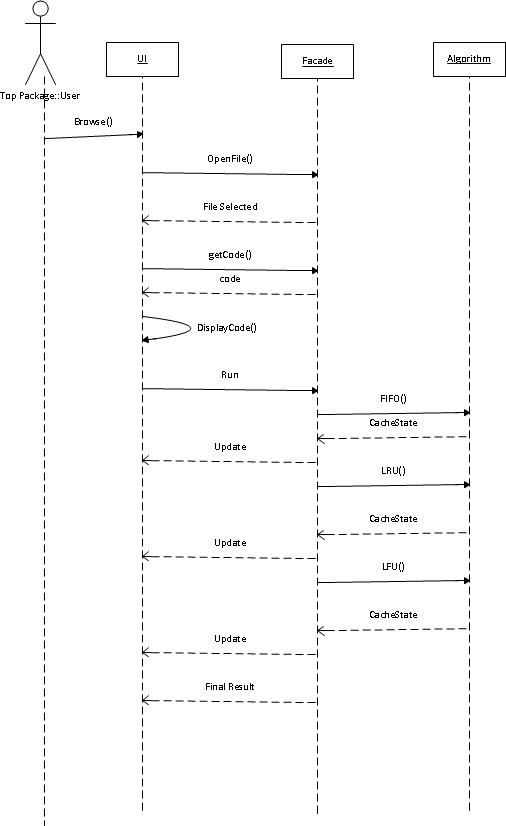
# PROGRAM STRCUTURE AND DIAGRAM

## Class Diagram



## Flow Of Program

The flow of program is implicated by the following given by the sequence diagram given below



# REFERENCES

* Program was developed using Microsoft Visual Studio 2010.
* Sequence and Class Diagrams were developed on Visio 2010
* Report was prepared using Microsoft Word 2010.
* Google Chrome and Google.com were used for internet assistance