Project Report on

# CACHE MEMORY: PERFORMANCE ISSUES

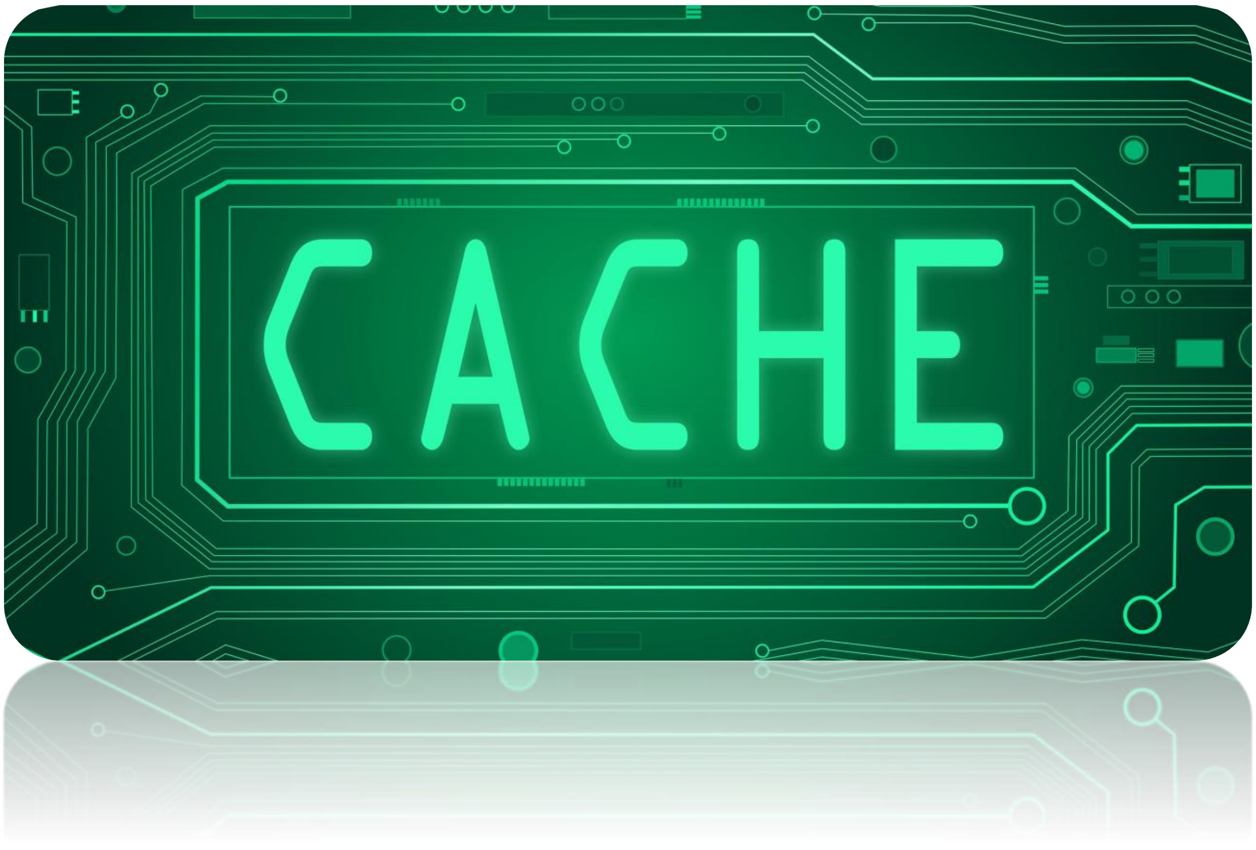
Submitted by:

**ANEESH PANCHAL - 2K20/MC/21**
**AYUSHI SAGAR - 2K20/MC/35**

Submitted to:

**MS. SUMEDHA SENIARAY**

Department of Applied Mathematics
Delhi Technological University



# Certificate

I hereby certify that the project dissertation titled “Cache Memory: Performance Issues” which is submitted by Aneesh Panchal (2K20/MC/21) and Ayushi Sagar (2K20/MC/35) of Mathematics and Computing Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this university or elsewhere.

**DELHI TECHNOLOGICAL UNIVERSITY**
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

## Acknowledgement

We, Aneesh Panchal(2K20/MC/21) and Ayushi Sagar (2K20/MC/35) would like to express our special thanks of gratitude to Ms. Sumedha Seniaray, Department of Applied Mathematics, Delhi Technological University for his able guidance and support in completing our project report.

We came to know about many new things, we are really thankful to him. We would also like to thank our classmates who have also helped whenever we were stuck at some point.

Thanking You

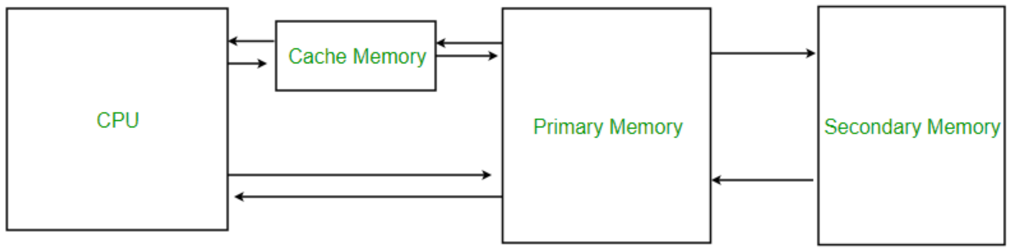
Aneesh Panchal (2K20/MC/21)
Ayushi Sagar (2K20/MC/35)

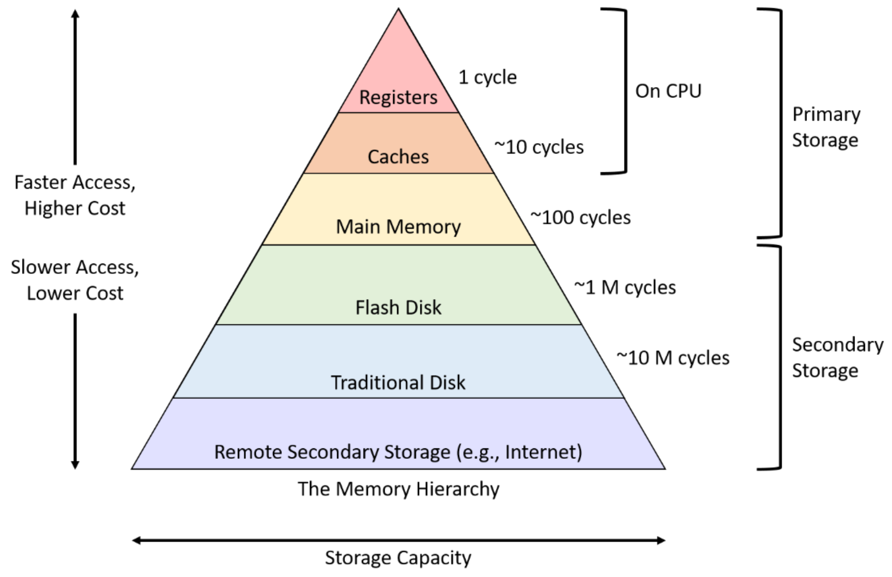
## Introduction:

For the project we have analyzed these research paper.

1. "Cache Memory: An Analysis on Performance Issue"
2. "Cache Memory: An Analysis on Optimization Techniques"
3. "Enhancing Cache Performance Based on Improved Average Access Time"
4. "Cache Memory: An Analysis on Replacement Algorithms and Optimization Techniques"
5. "Design Strategy of Cache Memory for Computer Performance Improvement"

Cache Memory is a very high speed memory that acts as a buffer between RAM and the CPU. It is used to speed up and synchronizing with high speed CPU. It is used to reduce the average time to access data from the main memory. Cache Memory is designed to combine the memory access time of expensive, high speed memory combined with large memory size of less expensive, lower speed memory.



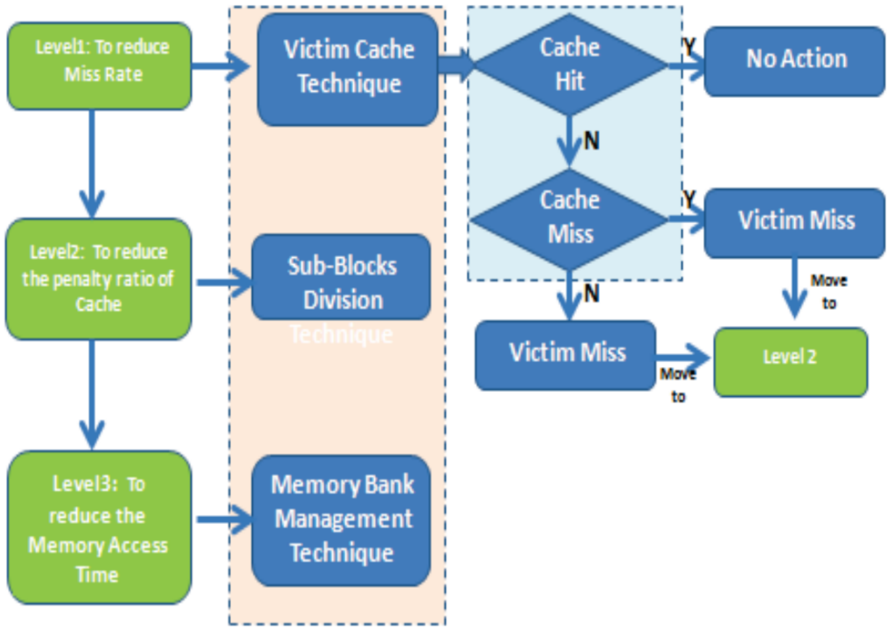


The Memory Hierarchy

## *Cache Memory: An Analysis on Performance Issue*

In this paper victim cache, sub-blocks, memory bank the three improvement techniques had been implemented simultaneously to increase the cache speed at each level. These techniques get implemented to improve and make the speed and performance of cache comparative to main memory. Also, the variables like miss penalty ratio, access speed of cache and miss rate ratio, had been used in this paper to estimate the cache memory performance after implementation of proposed approach.

In this architecture, all the above techniques had been combined together one by one, removing the limitations of previous one. This architecture will increase the speed of cache memory as each of the techniques discussed is helpful in improving the performance of the memory. Further, in the presented paper, different techniques and methodologies has been examined to assess the usefulness of proposed technology in improving performance.

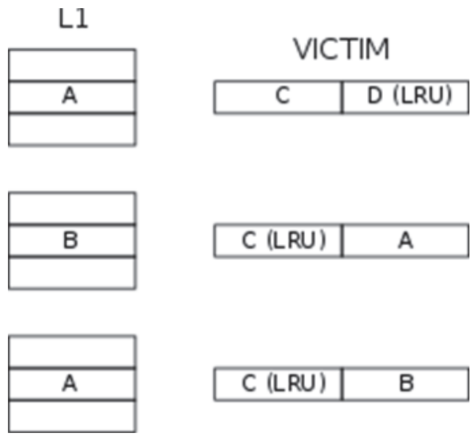

Level1: To reduce Miss Rate
Victim Cache Technique
Cache Hit
Y
No Action
N
Cache Miss
Y
Victim Miss
Move to
Level2: To reduce the penalty ratio of Cache
Sub-Blocks Division Technique
N
Victim Miss
Move to
Level 2
Level3: To reduce the Memory Access Time
Memory Bank Management Technique

### *Step 1: Reducing the Miss Rate using Victim Cache*

A victim cache also known as hardware cache memory, is used to boost hit latency rate for Direct-mapped cache memories and to reduce the conflict miss rate.

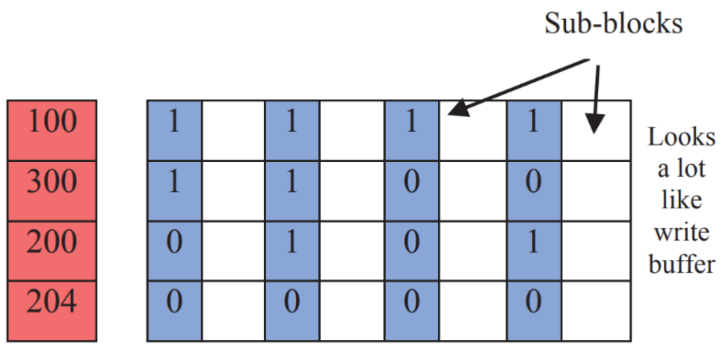
Let's consider a direct-mapped cache having two blocks A, B pointing towards the same set of values. Then it is considered to be linked to a second entry fully associative victim cache having C, D blocks.

The path to be considered, i.e. A, B, A, B…. and so on. It can be observed that when a victim cache hit occurs, part of blocks A and B will be swapped and the LRU block of victim cache does not change. Thus, we got an illusion of associatively towards the direct-mapped L1 cache which in contrast decreasing the number of conflict misses. If there are 2 cache memories, Level1 and Level2 with a particular policy (L2 and L1 do not store similar memory locations twice), L2 behaves like victim cache for L1.



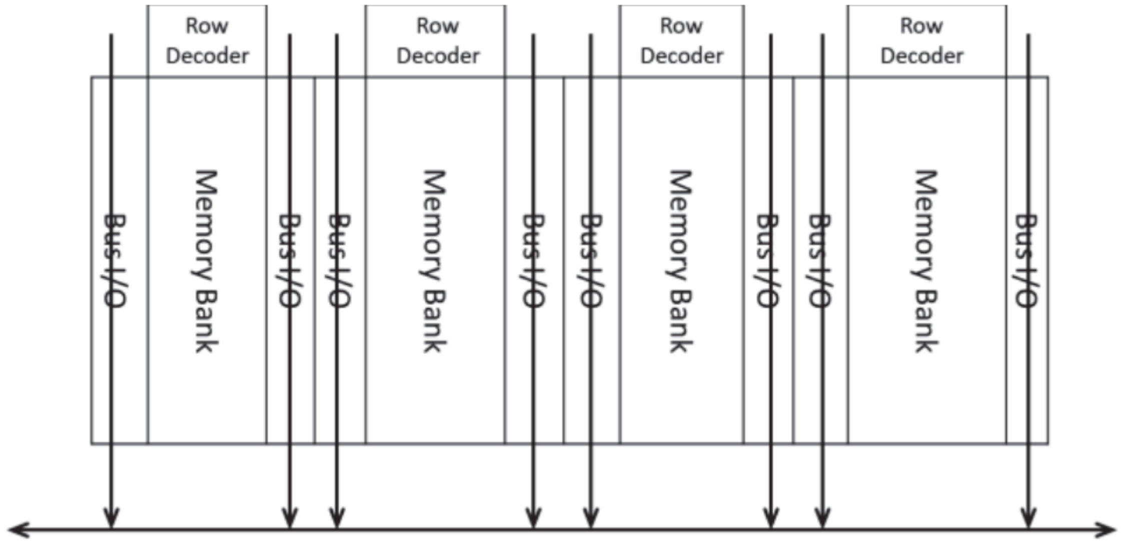
**_Step 2: Reducing the Penalty Ratio of Cache Miss_**

For the tags much space is required as a result the performance could be affected and hence cache speed is decreased. So we need to decrease the required space pf optimizing tags on the chip, large blocks are used. Due to reducing the necessary misses, the miss rate might be decreased too. However, because the full block should be transferred between cache and other memories, the miss penalty will be high. To overcome this problem, every block has to be divided into sub-blocks, each one of them has a valid bit. Although the tag is valid for the whole block, just a single block has to be read on a miss. So that a smaller miss penalty will be achieved since a block cannot be identified as the minimum unit fetched between cache and memory anymore.



***Step 3: Decreasing Memory Access Time***

A memory bank is a partition of cache memory that is addressed sequentially in the entire set of memory banks. For example, assuming that a(n) is data item which is stored in bank (b), the following data item a(n+1) will be stored in bank (b+1). To avoid the bank cycle time impacts, cache memory will be separated into several banks. Therefore, if data is saved or recovered sequentially each bank will have sufficient time to recover before processing the next request of the same bank.



**Conclusion:**

In the paper, a lot different methods are proposed on how can we improve performance of cache memory. The main conclusion that can be drawn from above paper is that we can decrease conflict miss rate by using bigger block size, but for this cache size is required.

In this work, different methods are proposed to improve the performance of cache memory. The main redemptions that we draw out from this study is that the conflict miss rate can be decreased after taking bigger block size, however for that, more cache size is required. Usage of bigger block size can rise penalty ratio of misses, decrease the time of the hit in addition to decreasing the power dissipation. Larger cache results in slow access time and more cost. However, more associatively results in fast access time and consequently less cycle time or lesser number of cycles. Victim Cache always decreases the rate of misses but at a higher cost in contrast to look aside miss cache.

One way of improving cache performance is to predict future access of data or instructions that need to be replaced from cache. When cache miss occurs data will be fetched from main memory and then it will be decided which data will be removed from the cache in order to place newly fetched data. For this purpose, we discuss performance of different algorithms for replacement methods. We discuss the performance of Least Recently Plus Five Least Frequently Replacement Policy and Blocked algorithm which reduces cache misses. The paper discusses cache structure, cache mapping techniques, strategies to reduce cache misses, techniques for avoiding cache penalties and attaining higher cache hit rates.

1. ***Replacement Algorithms***
   The less accessed cache sets utilization is increased by using replacement policy and programmable decoder. However, balanced cache consumes high power for each access comparing to direct mapped caches. Conflict misses can be reduced by saving blocks which are being replaced in victim cache. Victim caches have also been implemented in sectored cache which reduces cache misses. Less number of bit are used in sectored cache to store tags comparing to no-sectored cache. Optimum cache partition (OCP) is another replacement policy which uses cache hint along with LRU replacement policy.

2. ***Cache Algorithms***
   Cache-oblivious algorithm is used in CPU caches. Before evicting these cache lines one more check is placed on them which checks cache line for least frequently used out of those selected cache lines and then that with lowest frequency will be evicted or replaced. This algorithm has better performance than LRU. Block bypassing is also a technique to reduce miss rate of L1 cache. Block bypassing algorithm is also used for last level caches as their miss latency is higher than Level 1. It is based on feedback loops. It helps to reduce conflict misses.

3. ***Design Based Optimization***
   A user level technique had been implemented which is User Level Cache Control (ULCC) through which user can control the allocation of space in cache in their programs and hence reduce cache pollution.

A remapping module is used to allocate cache space by remapping between virtual and physical pages. This technique is very complex to implement and using this method produces less hit rates. Different methods are used to make cache's performance better. One of them is to make different banks of cache that can be accessed at different latencies.

## Conclusion:

In this paper, we discussed and evaluated the performance of different techniques and algorithms used for cache optimization. Through paper we can believe that it is hard to identify a certain cache optimization technique as the best choice in all the above cases. Each technique is associated with its design constraints, advantages and limitations. For example, the rate of conflict misses is reduced by using larger block size, larger cache and way prediction methods. However, using larger block size may increase miss penalty, reduced hit time and power consumption. On the other side, larger cache produces slow access time and high cost. It is associated with cache coherence problem. Higher associativity produces fast access time but they have low cycle time. Victim cache reduces miss rate at a high cost comparing to Cache miss look aside. LR + 5LF is a very good technique for reducing cache misses at a high rate comparing to some of the other techniques like LRU, FIFO, LFU. However, it is more complex than all of these methods. Cache pollution is eliminated by ULCC and also they have fast access time but complexity is high. Miss penalty is reduced by using pipelined cache which is very complex method. In future, we intend to enhance the multilevel cache by addressing its coherence issue.

## *Enhancing Cache Performance Based on Improved Average Access Time*

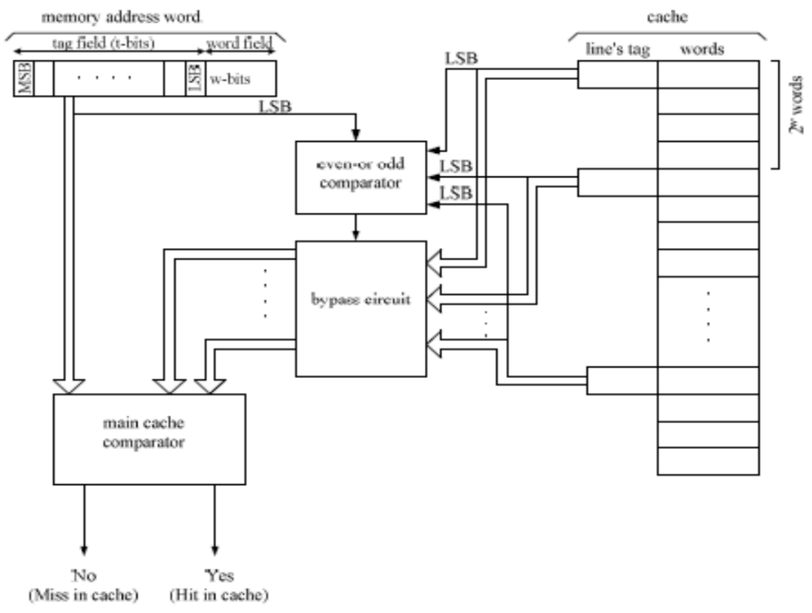
In this paper, we are optimizing the cache performance based on enhancing the cache hit ratio. The optimum cache performance is obtained by focusing on the cache hardware modification in the way to make a quick rejection to the missed line's tags from the hit –or miss comparison stage, and thus a low hit time for the wanted line in the cache is achieved.
In this paper, we explore the potential performance gains that cache conscious design offers in understanding and improving the performance. We develop a novel technique which we called Even-Odd Tabulation (EOT) to enhance the cache performance in terms of reducing the hit time.
The cache line's tags are tabulated into two groups: even line's tags and odd line's tags. Depending on the line number that is looking for. The cache lines tags of opposite LSB are undesired tags and rejected a way directly before going to the complete and long C – character comparison. By this approach, the line's tag of a missed mater, LSB does not pass to the C character comparison stage and there is no waste of time. Thus, the desired cache line is located quickly, leading to maximum hit ratio.

1. ***Direct Mapping***
   Direct mapped caches are fast, simple and inexpensive to implement. Moreover, direct mapped cache has been among the most popular cache architectures in the past and is still very common for off-chip caches. However, its main disadvantage is the frequent occurrence of conflict. Conflict occurs while executing a program task that requires several variables that resides in two or more blocks that map to the same cache line.

2. ***Fully Associative Mapping***
   The set associative mapping cache organization has most of the speed advantage of the direct mapping cache and much of the flexibility of the full associative cache, both at a moderate cost. Due to its cost-performance edge, set associative cache design has been selected by many computers manufactures when implementing a cache for their computer systems.

3. ***Sector Mapping Cache***

   When a block is cached, it is read into the correct position in any sector of the cache, given by discarding the bottom m address bits and taking the next n as the block number within the sector. That whole sector is then tagged with the remaining upper address bits and the other blocks in the sector are marked as invalid. This scheme takes advantage of locality of reference to consecutive blocks and needs fewer tags thus reducing the cost of associative access to the tags.



**Conclusion:**

In this paper, we presented optimum hardware cache architecture to enhance the performance of the cache based on high hit ratio. Because there are fewer cache lines than the main memory blocks, the cache line's tags can be come onto unequally two groups of tags; even line's tags and odd line's tags. So the proposed EOT approach exploited the LSB of the tag field in the main memory address to distinguish between the match tags and miss match tags in the cache. The even-or odd comparator compares only the LSBs of the cache line's tag and the memory field tag and rejects any miss match cache line in very low time. Consequently, only the matched cache line's tags are passed to the main cache comparator for a long and complete C-character comparison. In this way a lot of miss match line's tags are grasped and rejected quickly in the even-or odd comparator and they do not need to go to the main cache comparator. Building on that, there is no waste of time and a minimum time for cache line hitting is reached.

## *Cache Memory: An Analysis on Replacement Algorithms and Optimization Techniques*

In this paper, analysis is done on different cache optimization techniques as well as replacement algorithms. Furthermore, this paper presents a comprehensive statistical comparison of cache optimization techniques.

### *Replacement Algorithms*

1. *LRU (Least Recently Used) Algorithm*
   This algorithm discards the least recently used item from the cache in order to make space for the new data item.

2. *HLFU (History Least Recently Used) Algorithm*
   The HLFU algorithm will replace the cached objects based on Hist value as compared to the defined threshold in LFU.

3. *LFU (Least Frequently Used) Algorithm*
   This algorithm counts how often data items have been used.

4. *GDS (Greedy Dual Size)*
   In this algorithm index is calculated according to the size of a file. Larger the file smaller is the index.

5. *CAR (Clock with Adaptive Replacement) Algorithm*
   CAR is simple to implement and it has very low overhead on cache hits.

6. *ARC (Adaptive Replacement Cache) Algorithm*
   This algorithm continuously balanced recency and frequency features by responding to changing access pattern.

7. *RR (Random Replacement) Algorithm*
   This algorithm randomly selects any of the data item from the cache and replace it with the desire one.

8. *SLRU (Segmented LRU) Algorithm*
   This algorithm partitions the cache into two portions, one is unprotected and the other is protected.

**9.** *LR+5LF Algorithm*

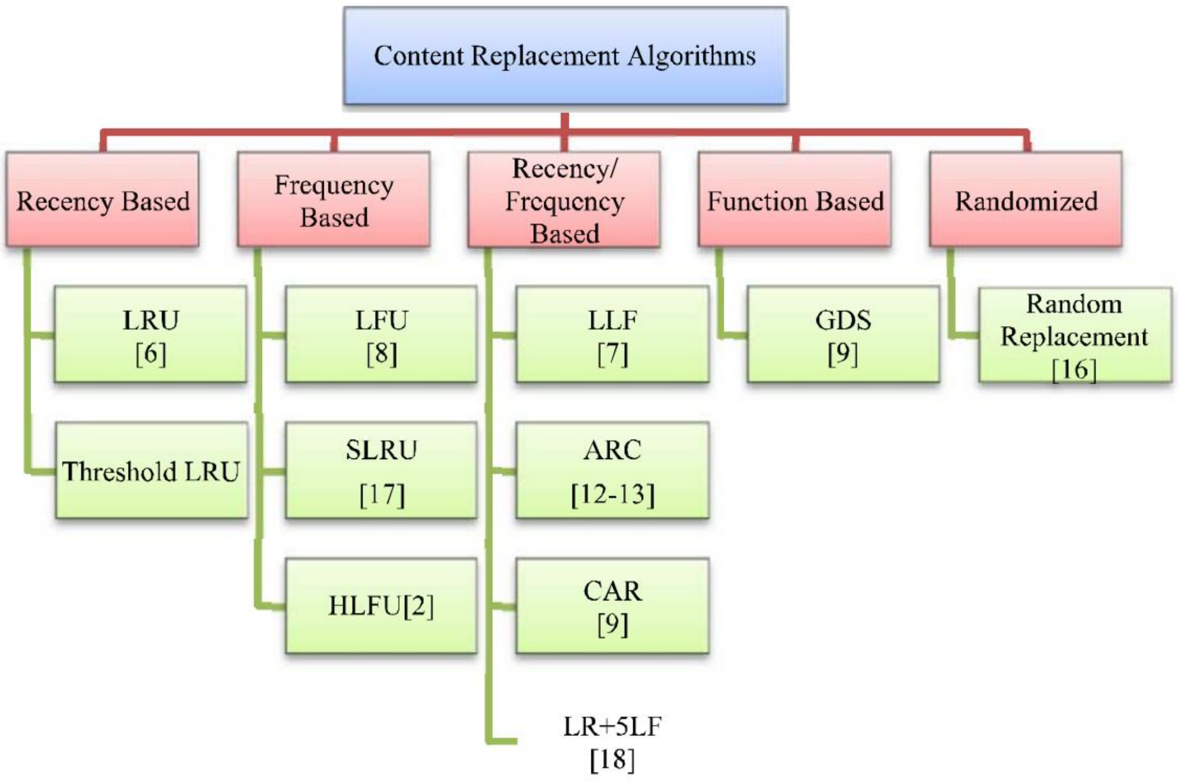
LR+5FU replacement policy is a combination of two popular replacement policies i.e. LRU and LFU.

**10.** *FIFO Algorithm*

The first in first out algorithm removes the page that has not been used for a long time.

**11.** *LLF (Lowest Latency First) Algorithm*

This algorithm keeps the average latency to a minimum by first expelling the object with the lowest download latency.



***Optimization Techniques***

One way to minimize the gap between memory latency and CPU cycle is the use of a multilevel cache. Miss rate of L1 (First Level) cache can be reduced by introducing L2 cache. Another technique called ULCC (User Level Cache Control) has been implemented through which user can control space allocation in cache.

This technique is hard to implement but it produces less hit rate and also reduce cache pollution. One of the other ways of cache optimization is compiler based optimization in which loops are optimizing through compiler. To set the accessed data in cache loops must be reduced to smaller size. In this way all the tasks will be executed consecutively which will be using same data from cache.

**Conclusion:**

By comparing different techniques we analyze that cache miss rate is reduced by using larger block size, larger cache, and way prediction method. Larger block size reduces hit time, increases miss penalty and consume high power. Higher associativity produces fast access time but they have low cycle time.

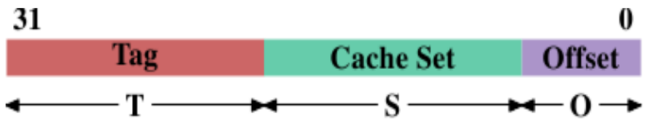
ULCC have fast access time and also used to eliminate cache pollution. Pipelined cache reduced miss penalty. Multilevel cache has very less miss penalty but yields high cycle time and have high power consumption.

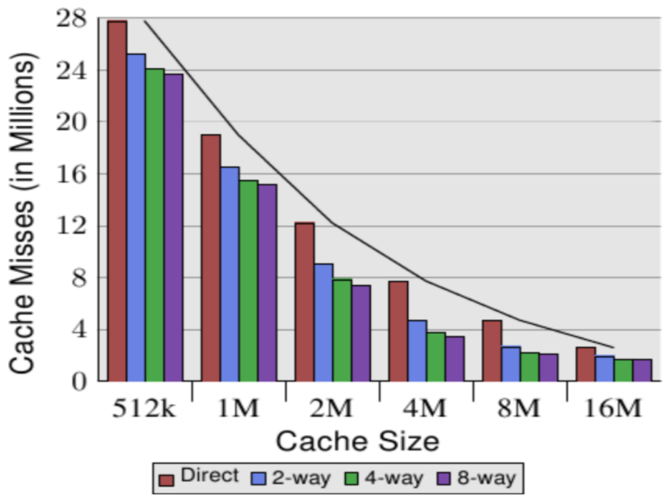
## *Design Strategy of Cache Memory for Computer Performance Improvement*

In this paper, it is tried to show how can the speed and performance can be improved. Following are the many factors that affect the performance if computer such as processor speed, the size of RAM and the weakness of the cache memory. Most cache memories are designed outside the processor units which are affecting the data transfer speed to/from the processor, delayed processor data access time, and processor access time. The simulation results explicit the great impact of the added cache memory on both the processor speed and the computer performance when the cache was designed inside the processor unit. Also, shows negative results when designing the cache outside the processor unit.

### *Cache Operation at High Level*

The entire cache line is loaded to L1d whenever memory content is needed by the processor. Let's consider a 64 bytes cache line so it means lower 6 bits are zero. The remaining bits are in some cases used to locate the line in the cache and as the tag.

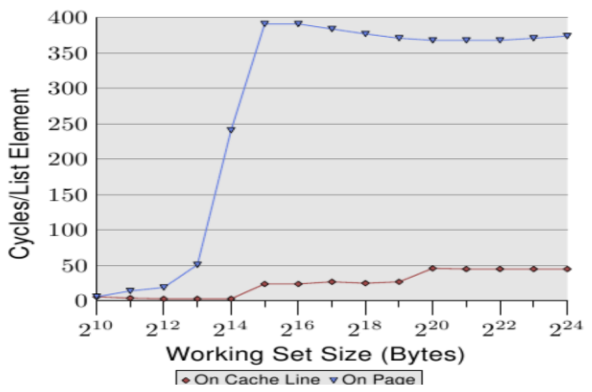


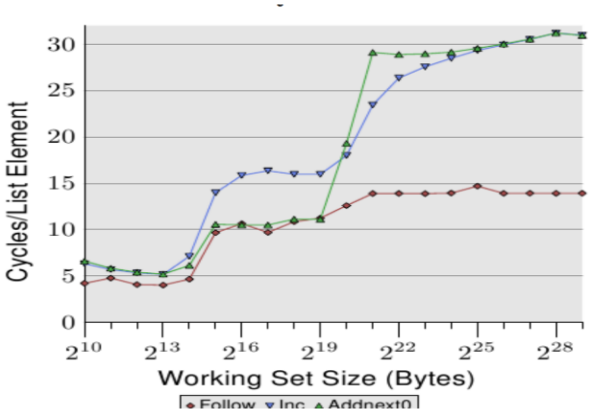


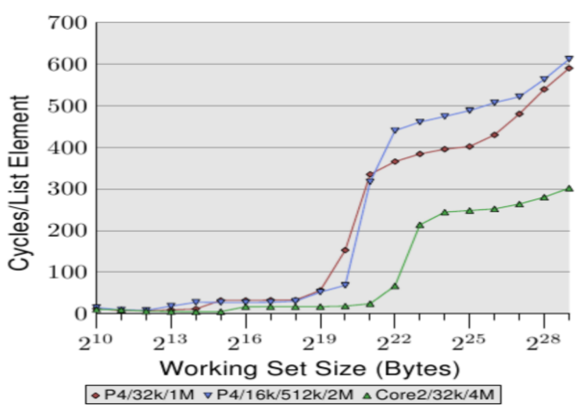
The graph states the data for a fixed cache line size of 32 bytes. While processing the graph it can be noted that associativity helps to reduce the number of cache misses. The processor can keep more of the working set in the cache with a set associative cache compared with a direct mapped cache.

Firstly, in the paper we have D-25 CONNECTOR which is used to interface the computer to the electronic circuit. It is used for device control and communication through software program. It consists of data, control, and status lines to be used as input/output buses. these lines are connected to relevant.
The lab link cable is connected to the computer for processing and display the output.
The 74245 is a high-speed Si-gate CMOS device. It is an octal transceiver featuring non-inverting 3- state bus compatible outputs in both send and receives directions.

***The Simulation Result***
The simulation process basically depends on normal parameters and RAM size in order to evaluate the effect of multi different cache levels on both processor speed and overall computer performance.







The main point of interest here is the region where the working set size is too large for the respective last level cache and the main memory gets heavily involved. And has shown that of the processor speed were obtained, the computer performance was improved when the cache was designed inside the processor unit, and the negative results when designing the cache outside the processor unit.

## ***References:***

- Sheshappa S.N., Ramakrishnan K.V. and G. Appa Rao, "*Enhancing Cache Performance Based on Improved Average Access Time*", International Journal of Scientific and Research Publications, Volume 2, Issue 11, November 2012 1, ISSN 2250-3153
- Sonia, Ahmad Alsharef, Pankaj Jain, Monika Arora, Syed Rameem Zabra and Gaurav Gupta, *"Cache Memory: An Analysis on Performance Issues", 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*
- Qaisar Javaid, Ayesha Zafar, Muhammad Awais and Munam Ali Shah, *"Cache Memory: An Analysis on Replacement Algorithms and Optimization Techniques"*
- Muhammad Waqas Ahmed and Munam Ali Shah, *"Cache Memory: An Analysis on Optimization Techniques"*, International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 04 – Issue 02, March 2015
- Tarig Ibrahim Osman Ahmed and Elsanosy M. Elamin, *"Design Strategy of Cache Memory for Computer Performance Improvement", International Journal of Research Studies in Electrical and Electronics Engineering(IJRSEEE) Volume 4, Issue 3, 2018, PP 12-17, ISSN 2454-9436(Online)*