Assignment I

Problem Bank 03

Assignment Description:

The assignment aims to provide deeper understanding of **cache by analysing its behaviour using cache implementation** CPU- OS Simulator. The assignment has three parts.

- Part I deals with Cache Memory Management with **Direct Mapping**
- Part II deals with Cache Memory Management with **Associative Mapping**
- Part III deals with Cache Memory Management with **Set Associative Mapping**

Assignment Set Number: **Problem Bank 03**

Group Name: **Group 323**

Contribution Table:

Contribution

Sl. No.	Name (as appears in Canvas)	ID NO	Contribution
1	RAGESH HAJELA	2021SC04877	100% Recorded the observations for all test cases, verified the values for other members, made graphs for all the tables, made observations for all the graphs
2	THAKARE KEDAR SATISHRAO	2021SC04878	100% Recorded the observations for all test cases, verified the values for other members, made graphs for all the tables, made observations for all the graphs
3	D V S RAMA KRISHNA	2021SC04879	100% Recorded the observations for all test cases, verified the values for other members, made graphs for all the tables, made observations for all the graphs

Resource for Part I, II and III:

- Used following link to login to “eLearn” portal.
  - https://elearn.bits-pilani.ac.in
- Click on “My Virtual Lab – CSIS”
- Used your canvas credentials login into Virtual lab
- In “BITS Pilani” Virtual lab click on “Resources”. Click on “Computer Organization and software systems” course.
  - Used resources within “LabCapsule3: Cache Memory”

Code used:

The following code written in STL Language was used, implements searching of an element (key) in an array using linear search technique.

```
program LinearSearch
        var a array(50) byte
        writeln("Array Elements: ")

        for n = 0 to 12
                a(n) = n
                writeln(a(n))
        next

        key = 11
        writeln("Key to be searched: ",key)

        found = 0

        for n = 0 to 12
                temp = a(n)
                if temp = key then
                        found = 1
                        writeln("Key Found",temp)
                        break
                end if
        next
        if found <> 1 then
                writeln("Key Not Found")
        end if
end
```

General procedure to convert the given STL program into ALP:

- Open CPU OS Simulator. Go to **advanced tab** and press **compiler** button
- Copy the above program in **Program Source** window
- Open **Compile** tab and press **compile** button
- In **Assembly Code,** enter **start address** and press **Load in Memory** button
- Now the assembly language program is available in CPU simulator.
- Set speed of execution to **FAST.**
- Open I/O console

- To run the program press **RUN** button.

General Procedure to use Cache set up in CPU-OS simulator

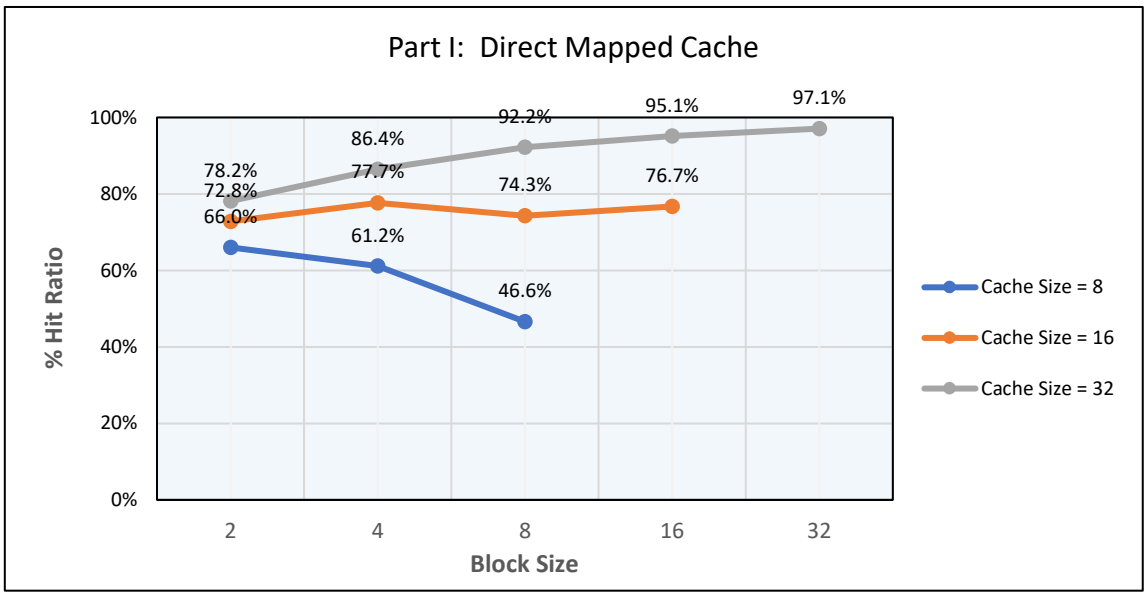
- After compiling and loading the assembly language code in CPU simulator, press "Cache-Pipeline" tab and select cache type as "both". Press "SHOW CACHE" button.
- In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm and write back policy.

Part I: Direct Mapped Cache

a. Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

Block Size	Cache size	# Hits	# Misses	% Miss Ratio	%Hit Ratio
2	8	136	70	33.9 %	66.1 %
4		126	80	38.8 %	61.2 %
8		96	110	53.4 %	46.6 %
2	16	150	56	27.1 %	72.9 %
4		160	46	22.3 %	77.7 %
8		153	53	25.7 %	74.3 %
16		158	48	23.3 %	76.7 %
2	32	161	45	21.8 %	78.2 %
4		178	28	13.5 %	86.5 %
8		190	16	7.8 %	92.2 %
16		196	10	4.8 %	95.2 %
32		200	06	2.9 %	97.1 %

b. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.



According to the values obtained and plotted on the graph, following observations are made:

Observation 1

i. It is evident from the data, higher the cache size, higher is the Hit ratio as observed for the same block size (say block size 2 in the table), with the increase in Cache size (say cache size 8, 16, 32), the no. of hits & hence hit ratio increases (136, 150, 161).
It is evident from the line graph where cache size = 8 with lowest hit ratio and the higher cache size i.e., 16 & 32 and above respectively.

Conclusion:

In Direct Mapping, each block of main memory maps to only one cache line. Higher the cache size, higher data can be brought into the Cache and make available for CPU. So, when CPU references for the data, there is higher probability of data available in the Cache results into higher Hit ratio. Hence higher the cache size, higher will be the %Hit ratio.
Also, if the program accesses 2 blocks which are mapped to the same line, results in higher misses.

Observation 2

i. Also, it is observed from the data, initially decreasing trend for Cache size 8 to flat trend where cache size is 16 and then increasing trend for cache size 32 with the increase in block size.

Conclusion:

Smaller blocks do not give advantage of spatial locality. Hence, increase in Cache size (i.e., for 8, 16, 32) results into higher Hit ratio with corresponding increase in block size.

c) Fill the below table and write a small note on your observation from the data **cache**.

- Block Size = 8
- Cache Size = 8
- Cache Type = Direct Mapped

Addresses	Data	Miss (%)
0120	**6F**	**0**
0121	**75**	**0**
0122	**6E**	**0**
0123	**64**	**0**
0124	**00**	**0**
0125	**00**	**0**
0126	**00**	**0**
0127	**00**	**0**

Observation 3

i. The %Miss Ratio is zero for this combination of Cache size = 8 and Block size = 8.

Conclusion:

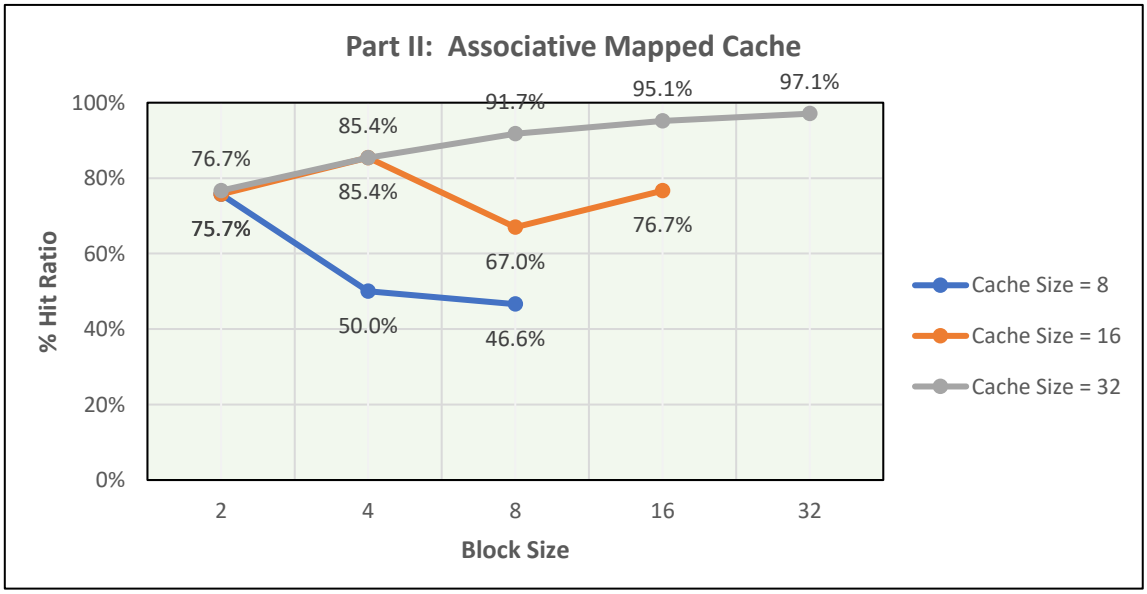
It implies that 100% Hit Ratio. When the Cache size is same as Block size, the complete block is loaded into the Cache resulting in 100% data is available in Cache for CPU. Hence the Hit ration is 100% while Miss ration is 0%.

Part II: Associative Mapped Cache

a. Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

LRU Replacement Algorithm					
Block Size	Cache size	# Hits	# Misses	% Miss Ratio	%Hit Ratio
2	8	156	50	24.2	75.8
4		103	103	50.7	49.3
8		96	110	53.4	46.6
2	16	156	50	24.3	75.7
4		176	30	14.6	85.4
8		138	68	33.0	67.0
16		158	48	23.3	76.7
2	32	158	48	23.3	76.7
4		176	30	14.6	85.4
8		189	17	8.3	91.7
16		196	10	4.9	95.1
32		200	6	2.9	97.1

b. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.



Observation 1

i. When compared with the Direct Mapping Method, in the Associative Mapping, the number of Hits & % Hit ratio is increased for the same Cache & Block size in each case (e.g., for Cache Size 8 Block size 2 in each case it is #hits are 136 and 156 respectively).

Conclusion:

In Associative Mapping, main memory block can be loaded into any of the cache line. This flexibility improves the Hit ratio. Higher Associative is lower miss rate. No of lines are more and can bring in more data.

Observation 2

i. Similarly to Direct Mapping, higher the cache size, higher is the Hit ratio as observed for the same block size (say 8 in the table), with the increase in Cache size (say 8, 16, 32), the no. of hits & hence hit ratio increases (96, 158, 200).

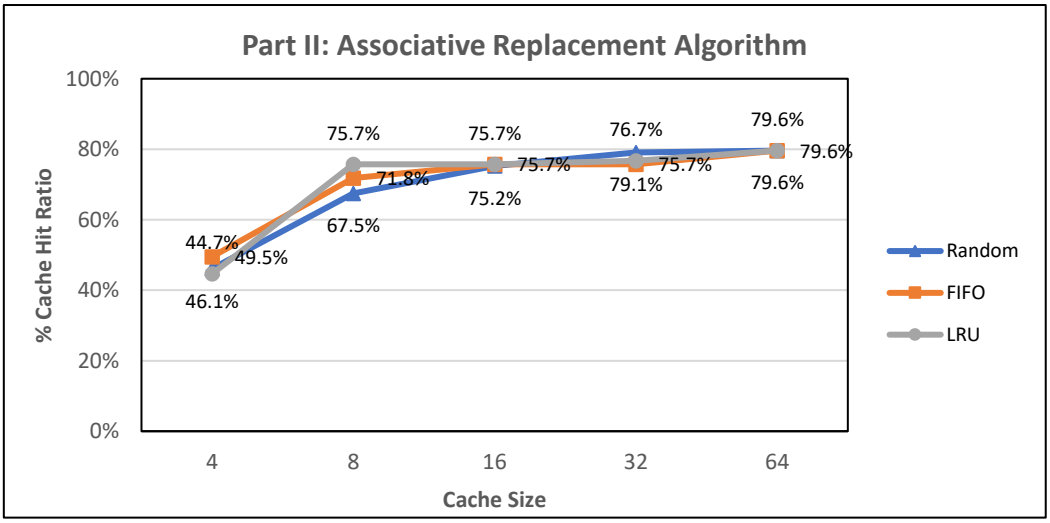
Conclusion:

Higher the cache size, higher data can be brought into the Cache and make available for CPU. So, when CPU references for the data, there is higher probability of data available in the Cache results into higher Hit ratio. Hence higher the cache size, higher will be the %Hit ratio.

c) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

Replacement Algorithm: Random				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	111	95	46.1%
2	8	67	139	67.5%
2	16	51	155	75.2%
2	32	43	163	79.1%
2	64	42	164	79.6%
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	104	102	49.5%
2	8	58	148	71.8%
2	16	50	156	75.7%
2	32	50	156	75.7%
2	64	42	164	79.6%
Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	114	92	44.7%
2	8	50	156	75.7%
2	16	50	156	75.7%
2	32	48	158	76.7%
2	64	42	164	79.6%

d) Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.



Observation 1

i. In Associative Mapping, in all the replacement algorithms (Random, FIFO & LRU), %Hit ratio increases with the cache size.

Conclusion:

Higher the cache size, higher data can be brought into the Cache and less replacements are required. Hence higher the cache size, higher will be the %Hit ratio irrespective of the Replacement algorithm.

Observation 2

i. The plot for different replacements algorithms is almost similar & differs a little.

Conclusion:

The little difference in the %Hit ratio which also depends on the program being executed and hence the plot for different replacement algorithm almost similar.

Part III: Set Associative Mapped Cache

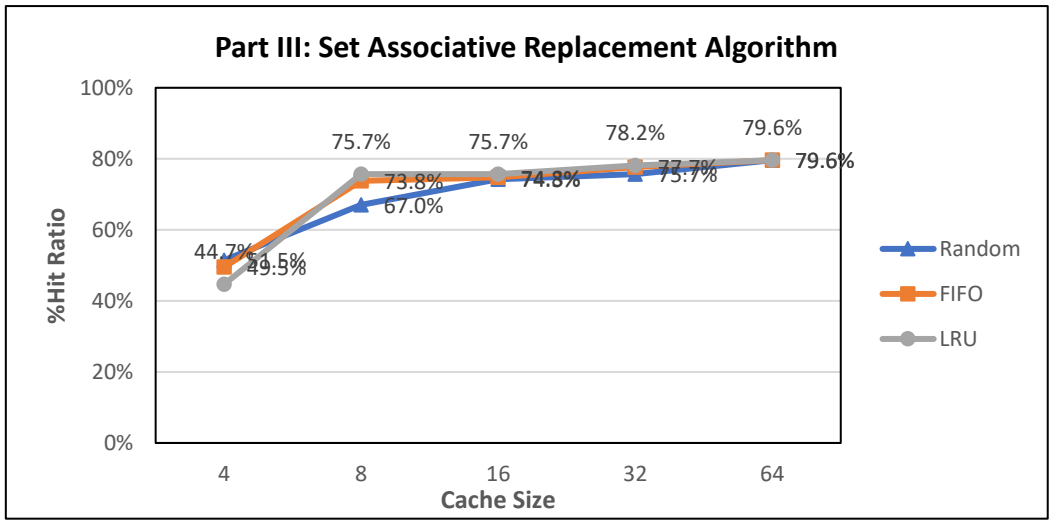
Execute the above program by setting the following Parameters:

- Number of sets (Set Blocks): 2 way
- Cache Type: Set Associative
- Replacement: LRU/FIFO/Random

a) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

Replacement Algorithm: Random				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	100	106	51.5%
2	8	68	138	67.0%
2	16	53	153	74.3%
2	32	50	156	75.7%
2	64	42	164	79.6%
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	104	102	49.5%
2	8	54	152	73.8%
2	16	52	154	74.8%
2	32	46	160	77.7%
2	64	42	164	79.6%
Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	114	92	44.7%
2	8	50	156	75.7%
2	16	50	156	75.7%
2	32	45	161	78.2%
2	64	42	164	79.6%

b) Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.



Observation 1

i. In Set Associative Mapping (2 Way) for Random, FIFO & LRU replacement algorithms, %Hit ratio increases with the cache size.

Conclusion:

Higher the cache size, higher data can be brought into the Cache and less replacements are required. Hence higher the cache size, higher will be the %Hit ratio irrespective of the Replacement algorithm.

Observation 2

i. Out of all the replacement algorithms, LRU gives better average Hit ratio when compared to Random & FIFO for different Cache.

Conclusion:

The Least Recently Used (LRU) is in the Cache for longest time without reference. Hence is most effective method of replacement algorithm give best Hit Ratio.

Also, the is little difference in the %Hit ratio which depends on the program being executed and hence the plot for different replacement algorithm almost similar.

Observation 3

i. In case of Random replacement algorithm, the #Hits & %Hit ratio varies from every program run.

Conclusion:

This difference in Hit ratio can be attributed to the Random Algorithm.

c) Fill in the following table and analyse the behaviour of Set Associative Cache. Which one is better and why?

Replacement Algorithm: LRU				
Block Size, Cache size	Set Blocks	Miss	Hit	Hit ratio
2, 64	2 – Way	42	164	79.6%
2, 64	4 – Way	42	164	79.6%
2, 64	8 – Way	42	164	79.6%

Observation 4

i. For 2-Way, 4-Way, 8-Way Set Associative LRU replacement algorithms, the Hit ratio is same i.e., 79.6%

Conclusion:

Higher the set size (way), higher is the cache lines per set, higher probability of data available in the Cache results into higher Hit ratio.

However, for higher set size (more cache lines per set), if there is a miss which requires to load the data from Main memory to Cache results in higher Average Memory Access Time.