Assignment I

Problem Bank 60

Assignment Description:

The assignment aims to provide deeper understanding of cache by analysing it's behaviour using cache implementation of 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

Submission: You will have to submit this documentation file and the name of the file should be GROUP-NUMBER.pdf. For Example, if your group number is 1, then the file name should be GROUP-1.pdf.

Submit the assignment by **7th June 2020, through canvas only**. File submitted by any means outside CANVAS will not be accepted and marked.

In case of any issues, please drop an email to the course TAs, Ms. Michelle Gonsalves (michelle.gonsalves@wilp.bits-pilani.ac.in).

Caution!!!

Assignments are designed for individual groups which may look similar and you may not notice minor changes in the assignments. Hence, refrain from copying or sharing documents with others. Any evidence of such practice will attract severe penalty.

Evaluation:

- The assignment carries 13 marks
- Grading will depend on
    - Contribution of each student in the implementation of the assignment
    - **Plagiarism or copying will result in -13 marks**

**************************FILL IN THE DETAILS GIVEN BELOW****************

Assignment Set Number: PROBLEM BANK 60

Group Name: COSS_ASSIGNMENT_GROUP_060

Contribution Table:

Contribution (This table should contain the list of all the students in the group. Clearly mention each student's contribution towards the assignment. Mention "No Contribution" in cases applicable.)

Sl. No.	Name (as appears in Canvas)	ID NO	Contribution
1	**Pothula Maruthi Chowdary**	**2019HC04288**	**100**
2	**Alla Jayadhar**	**2019HC04284**	**100**
3	**Ajit Mohan Pattanayak**	2019HC04286	**100**

Resource for Part I, II and III:

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

Code to be used:

The following code written in STL Language to count number of even numbers:

Program count_even

```
VAR a array(9) INTEGER
Var len byte
len = 7
a(0) = 12
a(1) = 7
a(2) = 6
a(3) = 14
a(4) = 23
a(5) = 15
a(6) = 17
a(7) = 3

writeln("Numbers: ")

for n = 0 to len
        write(a(n),"  ")
next
writeln("")

VAR x integer
VAR y integer
VAR m integer
VAR z integer
VAR count integer

count = 0
for i = 0 to len
        x = a(i)
        y = x/2
        m = y*2
        z = x-m
        if z = 0 then
        count = count + 1
        end if
next
write("Count of even numbers: ", count)

end
```

General procedure to convert the given STL program in to 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

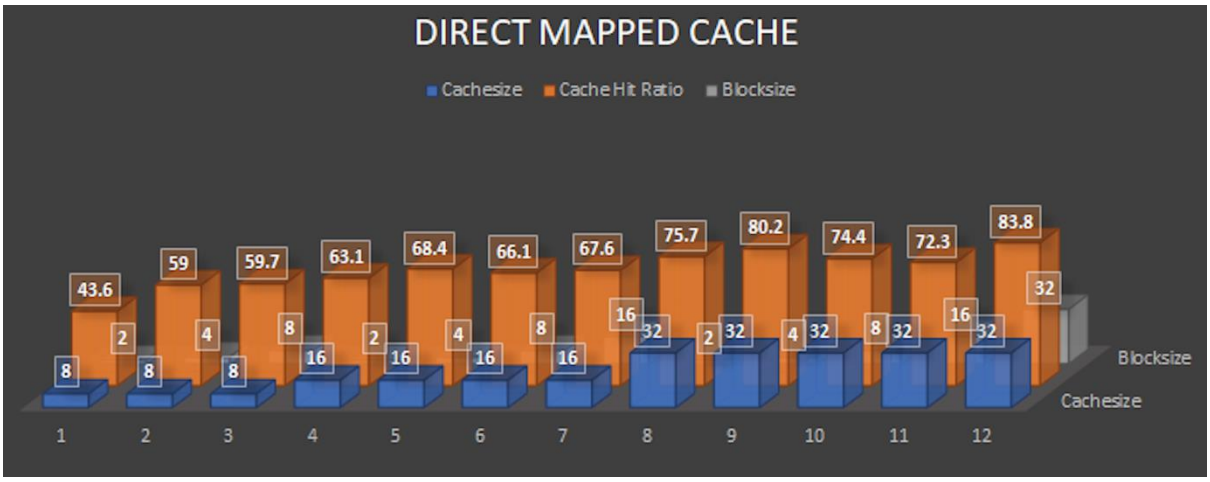
- After compiling and loading the assembly language code in CPU simulator, press "Cache-Pipeline" tab and select cache type as "data cache". Press "SHOW CACHE.." button.
- In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm and write policy.

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	205	263	56.2	43.6
4		276	192	41	59
8		279	189	40.3	59.7
2	16	295	173	36.9	63.1
4		320	148	31.6	68.4
8		309	159	33.9	66.1
16		316	152	32.4	67.6
2	32	354	114	24.3	75.7
4		375	93	19.8	80.2
8		348	120	25.6	74.4
16		338	130	27.7	72.3
32		392	76	16.2	83.8

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.



From the Above Diagram

1.It seems that HIT RATIO increases directly proportional to the block size accordingly when the Cache Size = 8
2.It seems that HIT Ratio fluctuates up and down but very little when the block size increases when the Cache Sizes are 16 and 32

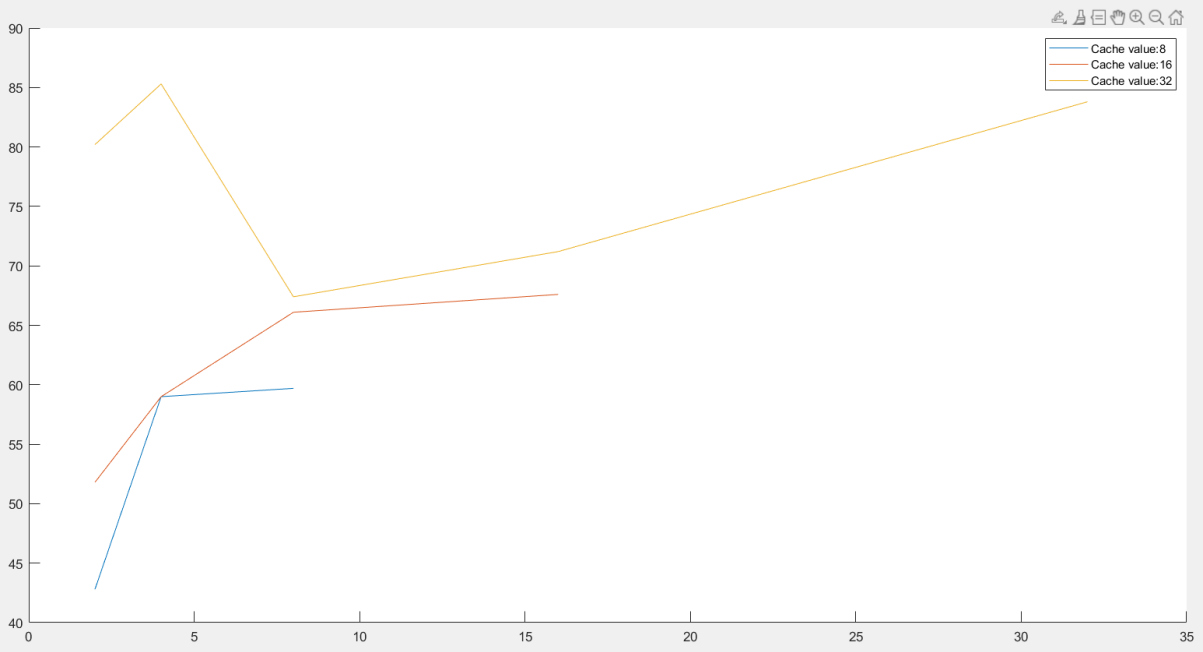
Over all increase in the HIT RATIO when the Block Size increase with respect to the Cache sizes.

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	200	268	57.2	42.8
4		276	192	41.0	59.0
8		279	189	40.3	59.7
2	16	242	226	48.2	51.8
4		276	192	41	59
8		309	159	33.9	66.1
16		316	152	32.4	67.6
2	32	375	93	19.8	80.2
4		399	69	14.7	85.3
8		315	153	32.6	67.4
16		333	135	28.8	71.2
32		392	76	16.2	83.8

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.



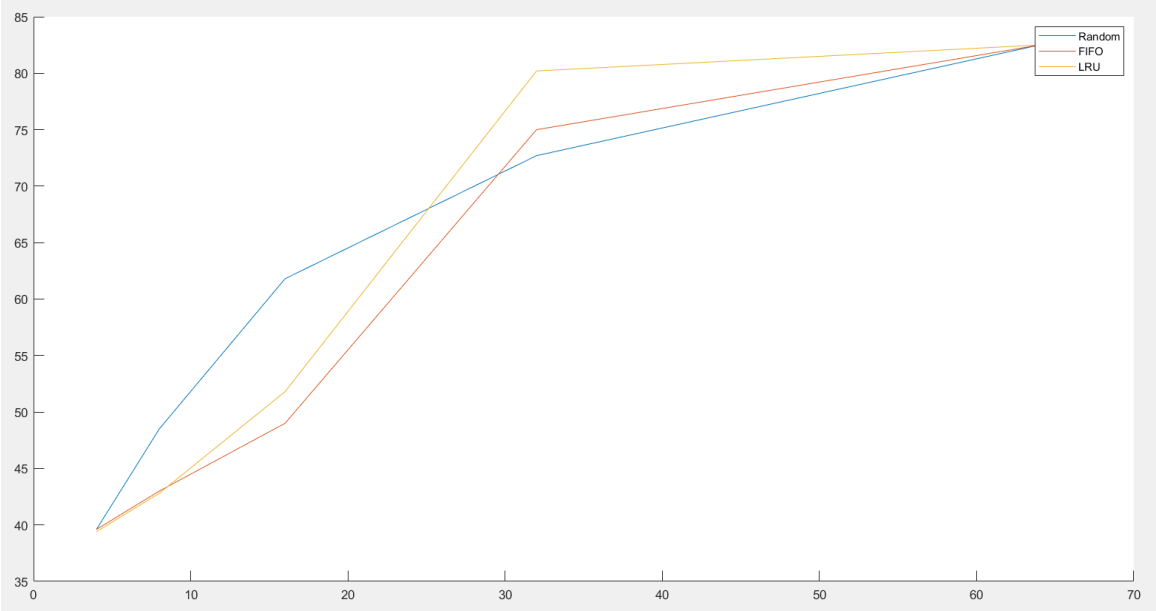
Few observations from the above graph, we are checking for the total of 468 blocks with different cache sizes. Here are few conclusions..

a. Hit ratio was increasing with increasing the cache size, which is expected because with increase in cache size, we can accommodate more blocks, hence the miss rate will be less.
b. There was sudden increase in the hit ratio if the block size was increased from 2 to 4 Bytes, this was due to spatial locality because in the above program after few instructions of initializations, the rest of the part we are running for ‘loop’ twice, we are using the registers which was recently used(in my case it was iterating between ‘R11’ and ‘R15’), so if these registers can able to readily available in cache , we can able to hit most of the times.
c. If we check the result at the cache size 32, there was a sudden drop at block size 8, we can observe two things from that,
    i. Though the hit ratio was less when compared to other block sizes of 32 cache size, still the value was high when compared with other cache size results which has block size of 8.
    ii. With increase in block size there will decrease of number of cache lines, hence at the very first instance we are not able to store the loop related registers.

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	283	185	39.6
2	8	241	227	48.5
2	16	179	289	61.8
2	32	128	340	72.7
2	64	82	386	82.5
Replacement Algorithm : FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	283	185	39.6
2	8	267	201	43
2	16	239	229	49
2	32	117	351	75
2	64	82	386	82.5
Replacement Algorithm : LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	284	184	39.4
2	8	268	200	42.8
2	16	226	242	51.8
2	32	93	375	80.2
2	64	82	386	82.5

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



Considering the three algorithms, **Random FIFO LRU, For the fixed block size Cache size and Hit Ratio are proportional and increased.**

From the observed behavior, LRU algorithm seems to have upper hand on FIFO for every block size and cache size, which is expected and theoretically proved. While random page replacement has slightly upper hand with block size 2 and less cache sizes because random replacement algorithm and Optimal algorithm was treated as an benchmark in two cases when we have lot of memory left and we are left with no memory. But the problem with random replacement was it was stack based algorithm, so because of Belady's anomaly,it's performance was decreased when compared to LRU.

Part III: Set Associative Mapped Cache

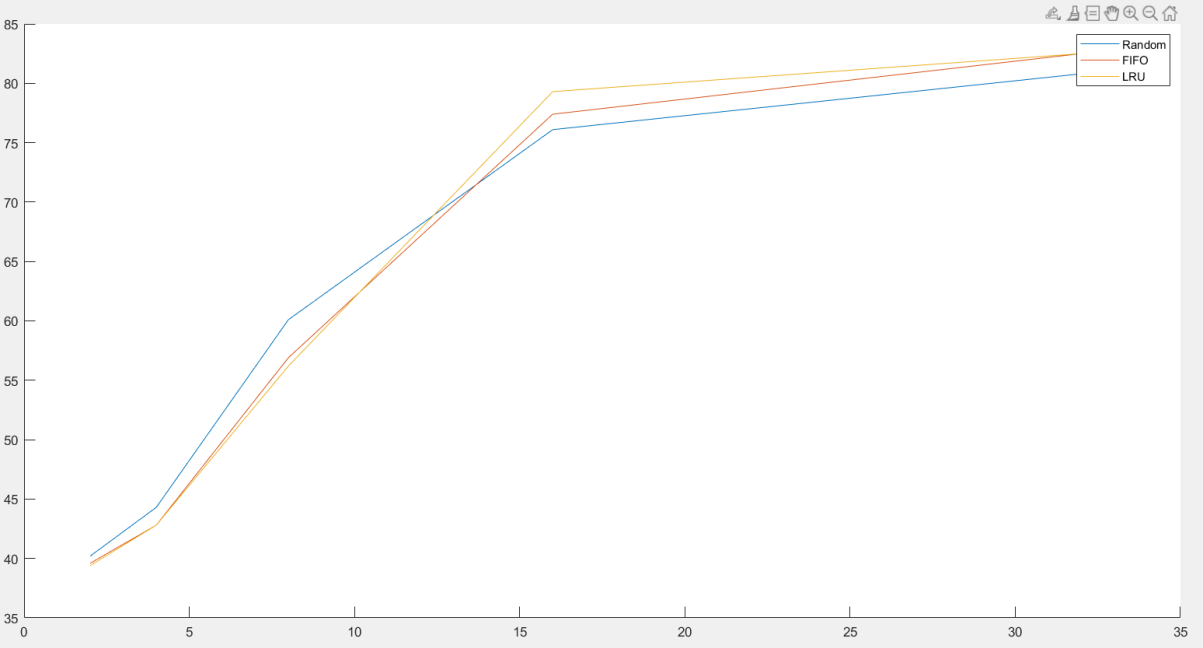
Execute the above program by setting the following Parameters:

- 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	280	188	40.2
2	8	261	207	44.3
2	16	187	281	60.1
2	32	112	356	76.1
2	64	90	378	80.8
Replacement Algorithm : FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	566	370	39.6
2	8	268	200	42.8
2	16	202	266	56.9
2	32	106	362	77.4
2	64	82	386	82.5
Replacement Algorithm : LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	284	184	39.4
2	8	268	200	42.8
2	16	205	263	56.2
2	32	97	371	79.3
2	64	82	386	82.5

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



Considering the different Replacement algorithms for set associative mapping , CacheHit ratio is increasing almost linearly based on the Cache size.

So Cache hit ratio is directly proportional to the Cache size . For all the three replacement algorithms.

From the observations , LRU seems to be the best one to select.

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

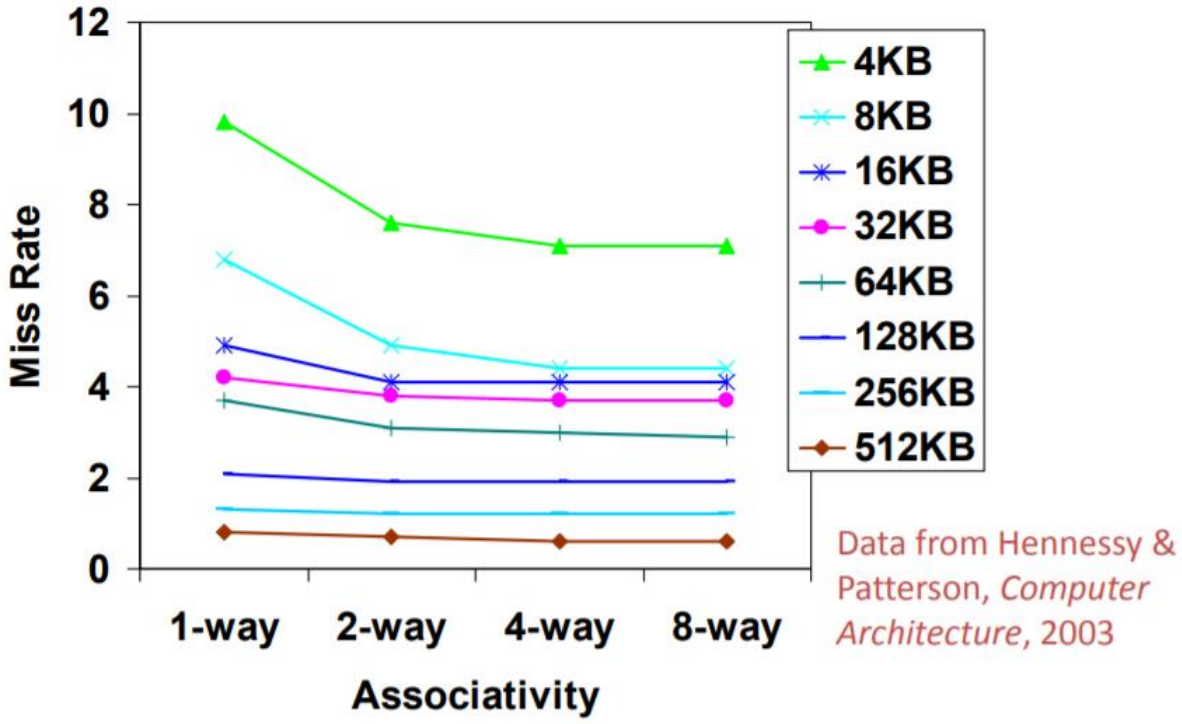
Replacement Algorithm : LRU				
Block Size, Cache size	Set Blocks	Miss	Hit	Hit ratio
2, 64	2 – Way	82	386	82.5
2, 64	4 – Way	82	386	82.5
2, 64	8 – Way	82	386	82.5

The reason for this result was,

a. More the catch size the effect of different mapping techniques to hit ratios will decrease. If you check the same by keeping the cache size as 32 bit then the result will be bit different, there will be slight decrease in hit ratio when we increase the mapping technique from 2-way to 8-way, this is because of decrease in number of options for the block replacement.

Block Size, Cache size	Set Blocks	Hit ratio
2, 16	2 – Way	56.2
2, 16	4 – Way	53
2, 16	8 – Way	51.8

Experimental results:



Source : https://www.ece.ucsb.edu/~strukov/ece154aFall2012/lecture15.pdf