Assignment I

Problem Bank 09

Assignment Description:

The assignment aims to provide deeper understanding of cache by analysing its 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.

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.

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. Remember that any kind of group changes after the announcement of the assignment is not allowed.

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 09

Group Name: Group 69

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.	**D M Pravin Kumar**	**2022DA04333**	**100%**
2.	**Deepak Kumar**		
3.	**Manindra Kumar**	**2022DA04310**	**100%**
4.	**Manish Kumar**	**2022DA04332**	**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 into 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, implements searching of an element (key) in an array using binary search technique.

```
program BinarySearch

VAR a array(13) INTEGER
for n = 0 to 12
    a(n) = n
    writeln (a(n))
next
VAR key INTEGER
VAR first INTEGER
VAR last INTEGER
VAR middle INTEGER
VAR temp INTEGER
key = 9
writeln("Key to be searched",key)

first = 0
last = 12
middle = (first+last)/2
while first <= last
    temp = a(middle)
    if temp = key then
        writeln("Key Found",middle)
        break
    end if
    if key > temp then
        first = middle + 1
    else
        last = middle - 1
    end if
    middle = (first+last)/2
wend
if first > last 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

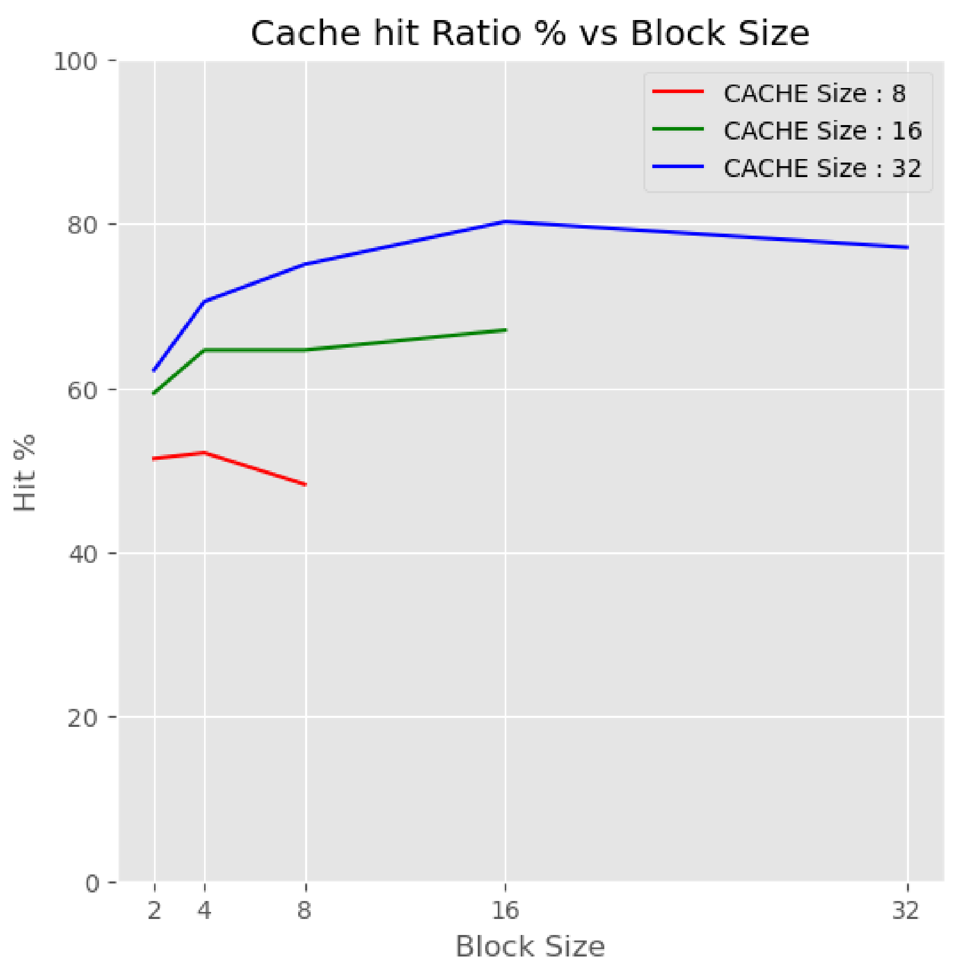
- After compiling and loading the assembly language code in CPU simulator, press "Cache-Pipeline" tab and select cache type as "both". Press "SHOW CACHE" button.
- In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm and write back policy.

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	148	140	48.61 %	51.39 %
4		150	138	47.92 %	52.08 %
8		139	149	51.74 %	48.26 %
2	16	171	117	40.63 %	59.38 %
4		186	102	35.42 %	64.58 %
8		186	102	35.42 %	64.58 %
16		193	95	32.99 %	67.01 %
2	32	179	109	37.85 %	62.15 %
4		203	85	29.51 %	70.49 %
8		216	72	25.00 %	75.00 %
16		231	57	19.79 %	80.21 %
32		222	66	22.92 %	77.08 %

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.



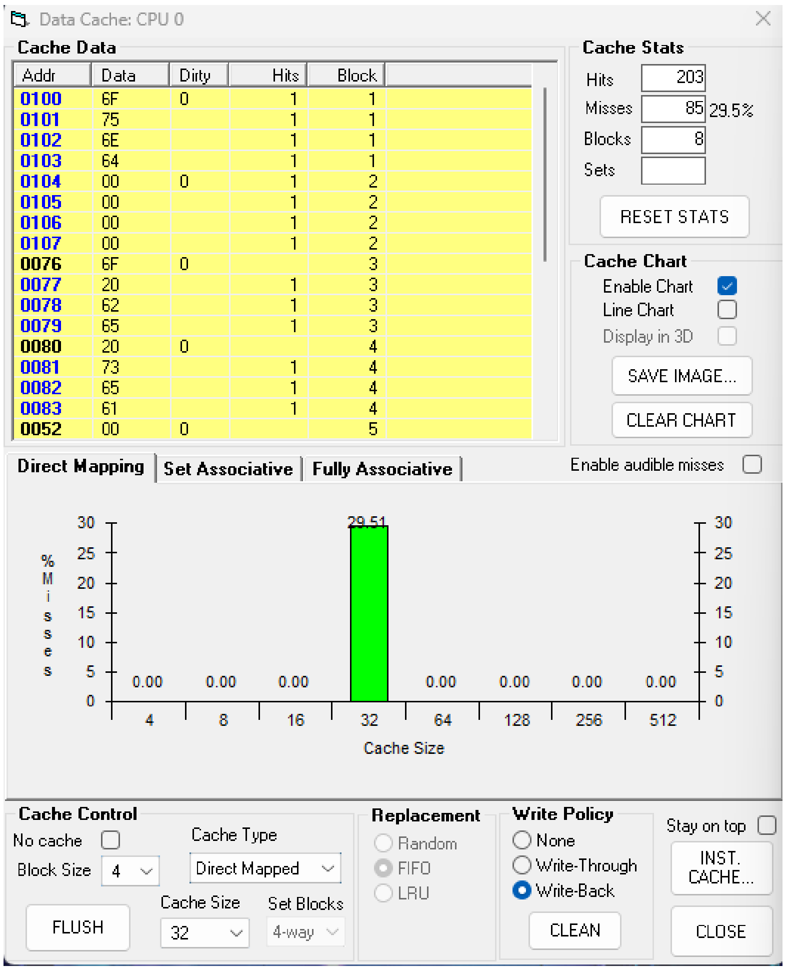
Comments:
As seen in above plot as the CACHE size increases the HIT ratio improves. For the same CACHE size as the block size increases the hi ratio increases first and then starts to decrease. Since for same cache size and block size we have 1 cache line which gives poor hit ratio. So at least 2 cache lines are preferred.

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

- Block Size = 4
- Cache Size = 32
- Cache Type = Direct Mapped

Addresses	**Data**	**Miss (%)**
0100	**6F**	**29.5 %**
0101	**75**	**29.5 %**
0102	**6E**	**29.5 %**
0103	**64**	**29.5 %**
0104	**00**	**29.5 %**
0105	**00**	**29.5 %**

0106	00	29.5 %
0107	00	29.5 %
0076	6F	29.5 %
0077	20	29.5 %
0078	62	29.5 %
0079	65	29.5 %
0080	20	29.5 %
0081	73	29.5 %
0082	65	29.5 %
0083	61	29.5 %
0052	00	29.5 %
0053	00	29.5 %
0054	00	29.5 %
0055	02	29.5 %
0056	00	29.5 %
0057	09	29.5 %
0058	02	29.5 %
0059	00	29.5 %
0060	07	29.5 %
0061	02	29.5 %
0062	00	29.5 %
0063	0C	29.5 %
0064	02	29.5 %
0065	00	29.5 %
0066	09	29.5 %
0067	02	29.5 %



Data Cache: CPU 0

Cache Data

Addr	Data	Dirty	Hits	Block
0100	6F	0	1	1
0101	75		1	1
0102	6E		1	1
0103	64		1	1
0104	00	0	1	2
0105	00		1	2
0106	00		1	2
0107	00		1	2
0076	6F	0		3
0077	20		1	3
0078	62		1	3
0079	65		1	3
0080	20	0		4
0081	73		1	4
0082	65		1	4
0083	61		1	4
0052	00	0		5

Cache Stats

Hits	203	
Misses	85	29.5%
Blocks	8	
Sets		

RESET STATS

Cache Chart

Enable Chart ✓
Line Chart
Display in 3D

SAVE IMAGE...
CLEAR CHART

Direct Mapping | Set Associative | Fully Associative

Enable audible misses

Cache Control

No cache
Block Size: 4
Cache Type: Direct Mapped
Cache Size: 32
Set Blocks: 4-way

FLUSH

Replacement

Random
[✓ FIFO]
LRU

Write Policy

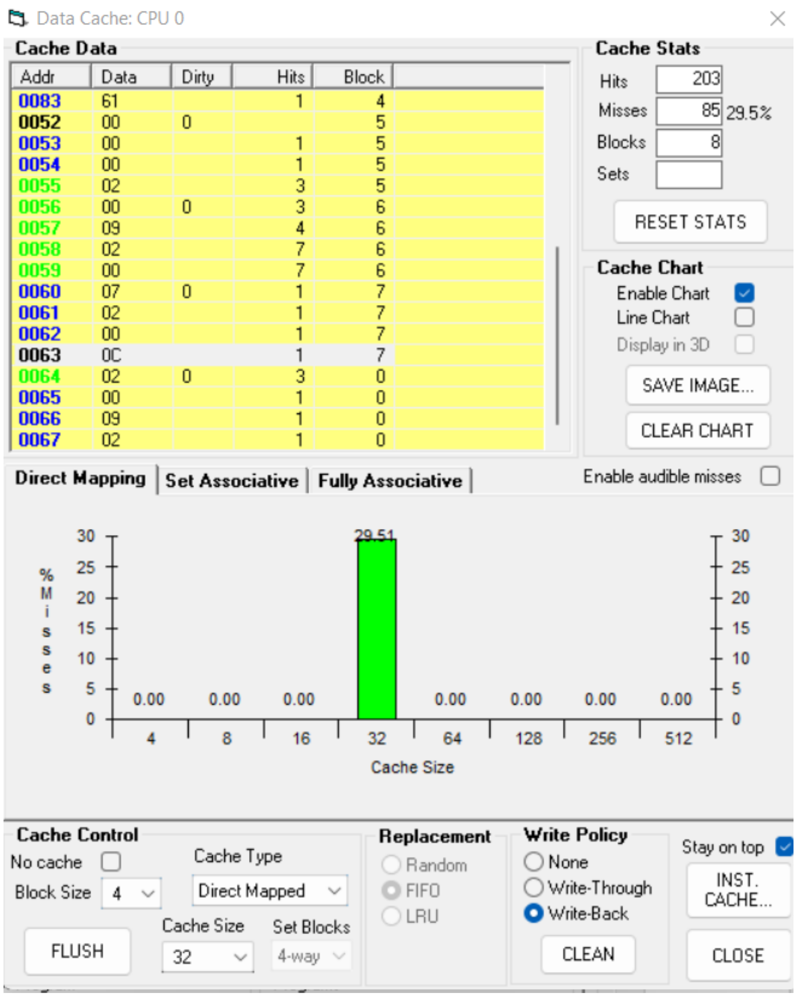
None
Write-Through
[✓ Write-Back]

CLEAN

Stay on top

INST. CACHE...

CLOSE


Data Cache: CPU 0

Cache Data

Addr	Data	Dirty	Hits	Block
0083	61		1	4
0052	00	0		5
0053	00		1	5
0054	00		1	5
0055	02		3	5
0056	00	0	3	6
0057	09		4	6
0058	02		7	6
0059	00		7	6
0060	07	0	1	7
0061	02		1	7
0062	00		1	7
0063	0C		1	7
0064	02	0	3	0
0065	00		1	0
0066	09		1	0
0067	02		1	0

Cache Stats

Hits 203
Misses 85 29.5%
Blocks 8
Sets

Cache Chart: Enable Chart [✓], Line Chart [ ], Display in 3D [ ]

Direct Mapping | Set Associative | Fully Associative

Enable audible misses [ ]

Cache Control: No cache [ ], Block Size 4, Cache Type Direct Mapped, Cache Size 32, Set Blocks 4-way

Replacement: Random, [✓ FIFO], LRU

Write Policy: None, Write-Through, [✓ Write-Back]

Stay on top [✓]

Comments -:

From the above screenshot we can see that there are only a few cache blocks which have most number of hits like block 6. This shows the limitation of the Data Cache. Since each block in main memory are mapped to a particular cache line in the data cache therefore if it happens that two most referred memory blocks in main memory are mapped to same cache line then we will not achieve the maximum efficiency from the Cache storage. System will again and again use the same cache line although some other Cache lines are available to use.

Part II: Associative Mapped Cache

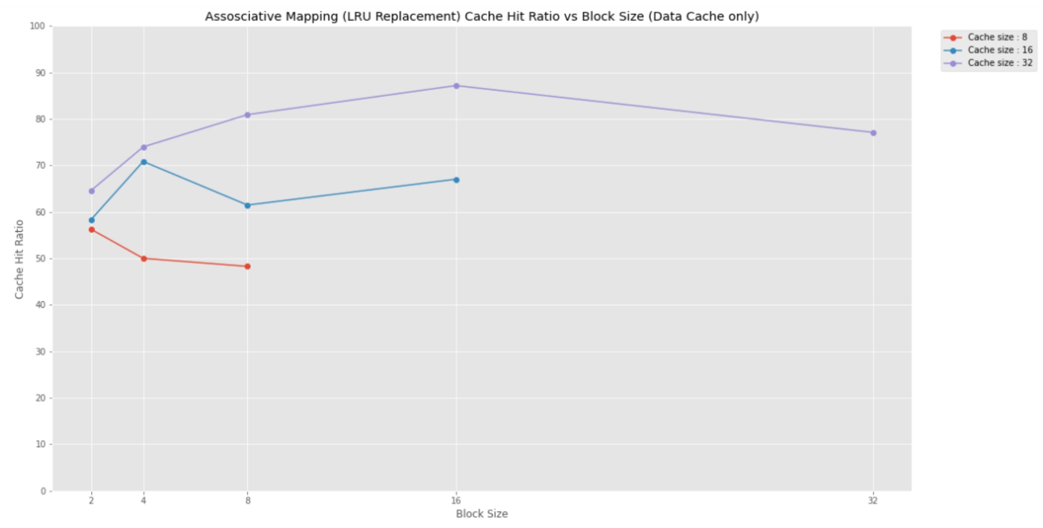
a) Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

LRU Replacement Algorithm					
Block Size	Cache size	# Hits	# Misses	% Miss Ratio	%Hit Ratio
2	8	162	126	43.75%	56.25%

4		144	144	50.00%	50.00%
8		139	149	51.74%	48.26%
2	16	168	120	41.67%	58.33%
4		204	84	29.17%	70.83%
8		177	111	38.54%	61.46%
16		193	95	32.99%	67.01%
2	32	186	102	35.42%	64.58%
4		213	75	26.04%	73.96%
8		233	55	19.10%	80.90%
16		251	37	12.85%	87.15%
32		222	66	22.92%	77.08%

Note -: The total number of hits and misses mentioned above are from Data Cache only.

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.



- The Higher Cache size is helping to increase the Hit Ratio, however in order to receive the highest Hit ratio we should have at least more than one Cache line. As per above graph we can see that the Hit ratio for the Cache of size 32 is better when Block size is 16 than when the block size is 32. When Block size is 32, we will have only one Cache line and that will yield us lower Hit Ratio. Therefore, it is always better to have higher Cache size but at the same time, it is always better to have at least 2 Cache lines.

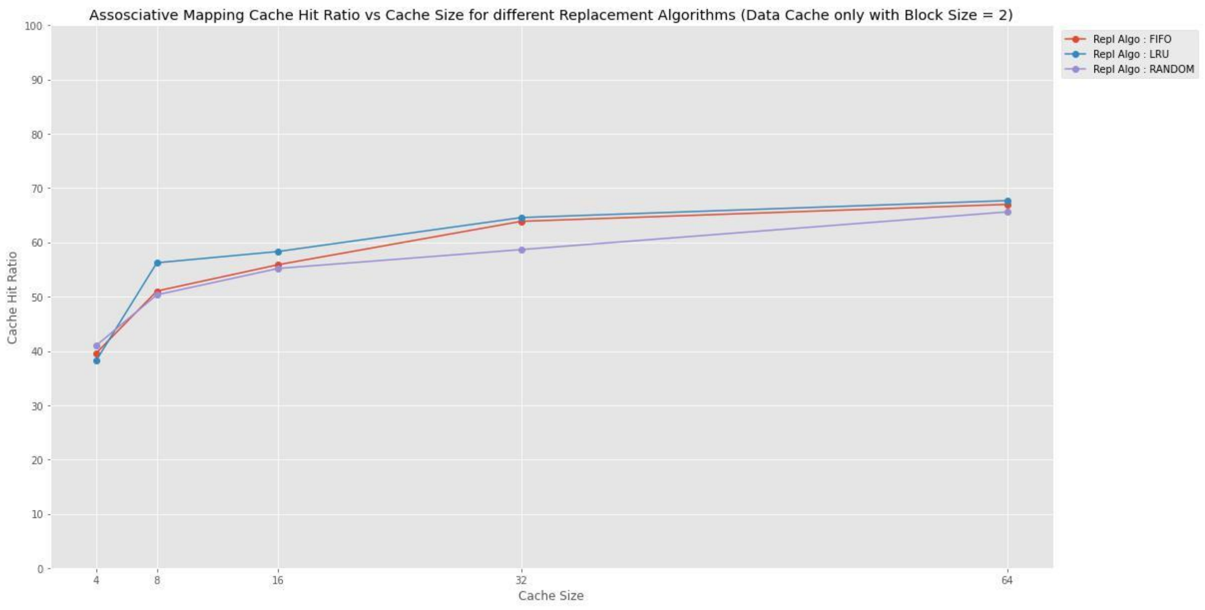
c) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

Replacement Algorithm: Random				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	170	118	40.97%
2	8	143	145	50.35%
2	16	129	159	55.21%
2	32	119	169	58.68%
2	64	99	189	65.63%
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	174	114	39.58%
2	8	141	147	51.04%
2	16	127	161	55.90%
2	32	104	184	63.89%
2	64	95	193	67.01%
Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	178	110	38.19%
2	8	126	162	56.25%
2	16	120	168	58.33%
2	32	102	186	64.58%

2	64	93	195	67.71%

- Looking at the above data, we can say that the LRU replacement algorithm is yielding the best Hit ratio with a higher Cache size. This algorithm keeps a track of the Cache line which was least recently used and therefore is very effective in detecting the Cache line to be used. This also ensures that all the Cache Lines are properly utilized and eliminates the chances of having a particular Cache line not being used for a long time.

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



- According to above graph, we can say that with lower Cache size, the LRU algorithm yields the best Hit ratio when the Block size is lower. However, as we increase the Cache size, all the replacement algorithms are giving same Hit Ratio.

Part III: Set Associative Mapped Cache

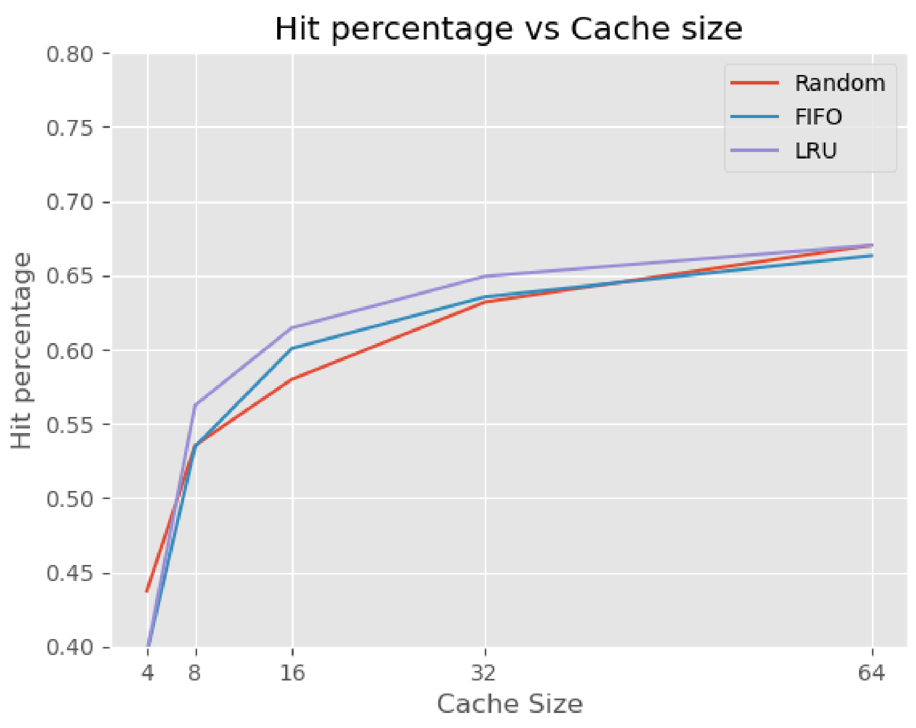
Execute the above program by setting the following Parameters:

- 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	162	126	43.75%
2	8	137	151	52.43%
2	16	121	167	57.99%
2	32	106	182	63.19%
2	64	95	193	67.01%
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	174	114	39.58%
2	8	134	154	53.47%
2	16	115	174	60.07%
2	32	105	183	63.54%
2	64	97	191	66.32%
Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	174	114	39.58%
2	8	126	162	56.25%
2	16	111	177	61.46%
2	32	101	187	64.93%
2	64	95	193	67.01%

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



From above plot LRU has slightly better hit ratio than random and FIFO for given experiment. Random replacement is having worst hit ratio among these.

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	95	193	67.01
2, 64	4 – Way	95	193	67.01
2, 64	8 – Way	97	191	66.31

With increase in set blocks the Hit ratio decreases. With increase in Set blocks the number of sets are getting lower like for a 8-way Set associative Cache for Block size 2 and Cache size 64, the total number of Cache lines will be 32 and with 8 lines in each set, we will have 4 sets in this case. When the Cache is 4 way, we will have total 8 sets in the Cache and hence we are seeing an increase in Hit Ratio with increase in the number of Sets.

So lesser set blocks are better from above example.