Assignment I

Problem Bank 14

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:

Group Name:

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	BHARATH S	2022da04409	100%
2	CHANDUPATLA ANIRUDH REDDY	2022da04387	100%
3	DISYA RAKSHITA S	2022da04467	100%
4	MEDAPATI JAYANAGA SURESH REDDY	2022da04445	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 Sorting of elements in an array using Bubble Sort technique.

```
program BubbleSort

var a array(6) byte
a(0) = 8
a(1) = 5
a(2) = 9
a(3) = 2
a(4) = 6
a(5) = 3
var len byte
var temp byte
var l1 byte
var l2 byte
var x1 byte
var x2 byte
var j byte
var j1 byte
var k byte
var i byte
len = 6
l1 = len - 1
for k =0 to len
        write(a(k),"  ")
next
writeln("")
writeln("Bubble Sort Starts")
for i = 0 to l1
        l2 = len - i - 1
        for j=0 to l2
                j1 = j + 1
                x1 = a(j)
                x2 = a(j1)
                if x1 > x2 then
                temp = a(j1)
                a(j1) = a(j)
                a(j) = temp
                end if
        next
        for k =0 to len
                write(a(k),"  ")
        next
        writeln("")
next
writeln("Bubble Sort Ends")
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 the CPU simulator.
- Set speed of execution to **FAST.**
- Open I/O console
- To run the program, press the RUN button.

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

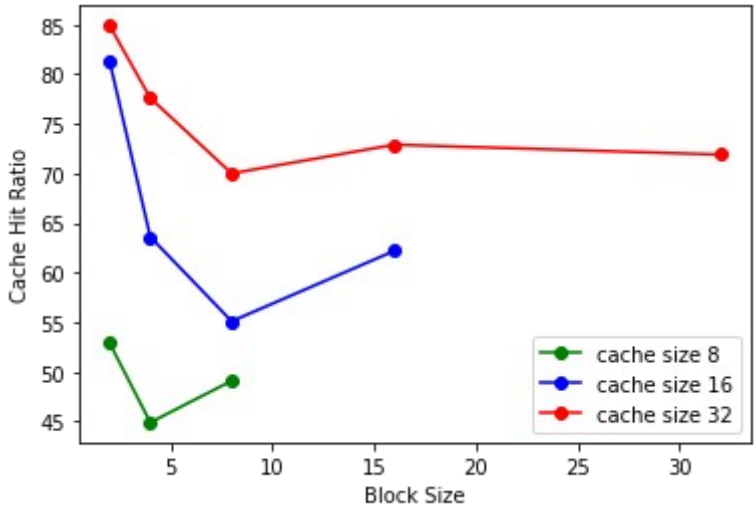
- After compiling and loading the assembly language code in the CPU simulator, press the “Cache-Pipeline” tab and select the cache type as “both”. Press “SHOW CACHE.” button.
- In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm and write policy.

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	869	775	47.1	52.9
4		738	906	55.1	44.9
8		806	838	50.9	49.1
2	16	1172	472	28.7	81.3
4		1045	599	36.4	63.6
8		905	739	44.9	55.1
16		1022	622	37.8	62.2
2	32	1395	249	15.1	84.9
4		1275	369	22.4	77.6
8		1150	494	30.0	70.0
16		1197	447	27.1	72.9
32		1182	462	28.1	71.9

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.



- Increase in Block size didn't improve HIT Ratio
- Smallest Block size gave better HIT Ratios for all respective cache sizes
- HIT Ratio didn’t changed much between block size 16 & 32 for cache size 32

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

- Block Size = 4
- Cache Size = 8
- Cache Type = Direct Mapped

Addresses	Data	Miss (%)

Cache Data

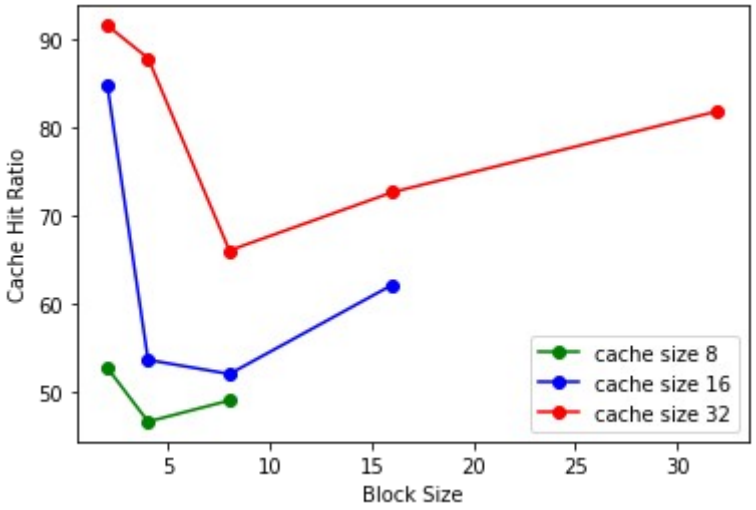
Addr	Data	Dirty	Hits	Block
0116	73	0	1	1
0117	00		1	1
0118	00		1	1
0119	00		1	1
0112	20	0	1	0
0113	45		1	0
0114	6E		1	0
0115	64		1	0

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	868	776	46.6	52.8
4		702	942	57.3	46.7
8		806	838	50.9	49.1
2	16	1393	251	15.2	84.8
4		882	762	46.3	53.7
8		855	789	47.9	52.1
16		1022	622	37.8	62.2
2	32	1506	138	8.4	91.6
4		1444	200	12.1	87.9
8		1086	558	33.9	66.1
16		1194	450	27.3	72.7
32		1182	462	28.1	81.9

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.



- Increase in Block size didn't improve HIT Ratio
- Smallest Block size gave better HIT Ratios for all respective cache sizes
- There is steady drop in HIT Ratio after block size 2 then again increased gradually after block size 4 for all cache sizes except for cache size 16.

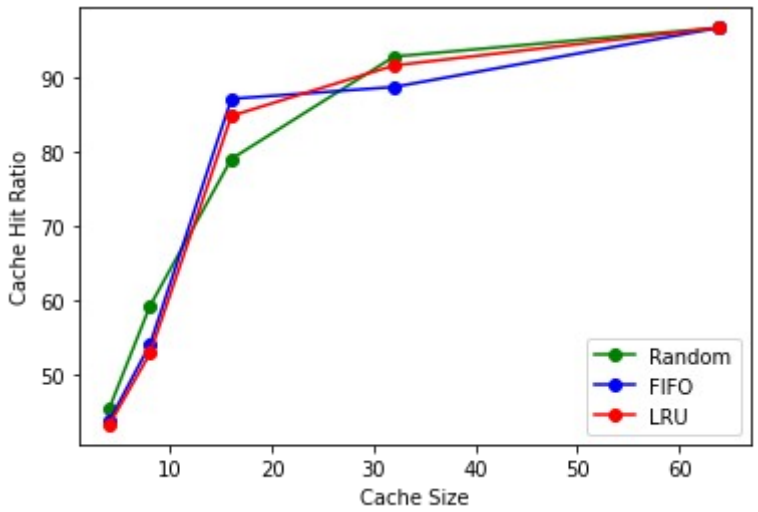
c) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

Replacement Algorithm: Random				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	899	745	45.4
2	8	671	973	59.2
2	16	346	1298	79.0
2	32	118	1526	92.8
2	64	54	1590	96.7
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	926	718	43.7
2	8	757	887	54.0
2	16	262	1382	87.1
2	32	187	1457	88.7
2	64	54	1590	96.7

Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	951	693	43.2
2	8	776	868	52.8
2	16	251	1393	84.8
2	32	138	1506	91.6
2	64	54	1590	96.7

- for cache size 64 all three replacement algorithms gave same HIT Ratio
- for cache size 4,8 & 32 Random algorithm gave the best HIT Ratio
- for cache size 16 FIFO algorithm gave the best HIT Ratio
- conclusion : The best is Random algorithm since it gave the best HIT Ratio for majority of cache size setting

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



- For all three (i.e Random, FIFO, LRU) algorithms HIT Ratio increased steadly for cache sizes (4,8,16) but hereafter only for Random and LRU there is steady increase in HIT Ratio against cache sizes (32,64) but for FIFO there is one dip in HIT Ratio at cache size 32

Part III: Set Associative Mapped Cache

Execute the above program by setting the following Parameters:

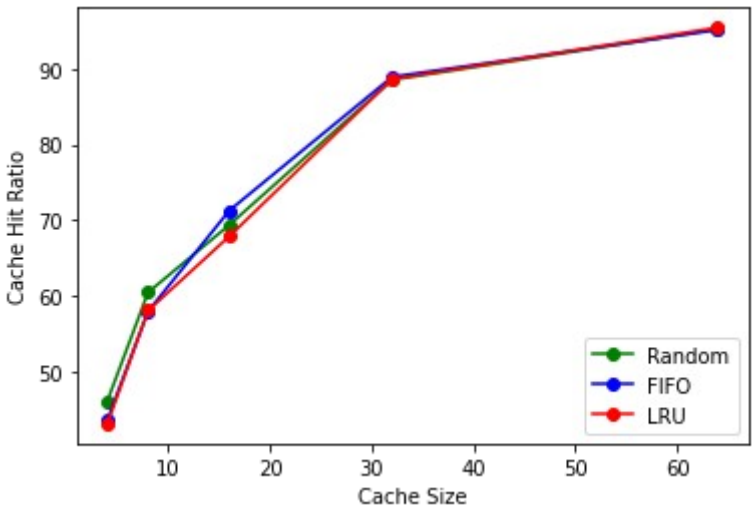
- Number of sets (Set Blocks): 2 way
- Cache Type: Set Associative
- Replacement: LRU/FIFO/Random

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	886	758	46.2
2	8	649	995	60.5
2	16	504	1140	69.4
2	32	190	1454	88.5
2	64	80	1564	95.2
Replacement Algorithm: FIFO				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	926	718	43.7
2	8	691	953	58.0
2	16	472	1172	71.3
2	32	184	1460	88.9
2	64	82	1562	95.1
Replacement Algorithm: LRU				
Block Size	Cache size	Miss	Hit	Hit ratio
2	4	951	693	43.2
2	8	690	954	58.1
2	16	529	1115	67.9
2	32	188	1456	88.6
2	64	76	1568	95.4

- Random algorithm gave the best HIT Ratio for cache size 4 & 8
- FIFO algorithm gave the best HIT Ratio for cache size 16 & 32
- LRU algorithm gave the best HIT Ratio for cache size 64
- conclusion: there is tie between Random and FIFO

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



- Cache hit ratio is increasing steadly along with increase in Cache Size for all three replacement algorithms

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	76	1568	95.4
2, 64	4 – Way	87	1557	94.8
2, 64	8 – Way	61	1583	96.3

- As per this execution run 8-way set associative cache is better since it has hit ratio of 96.3% which is highest when compared with other two