It is my kind request to all of you please do a little research by yourself on following topics:
1. Why SV is used in Verification?
2. What is difference between SV & UVM?
3. Which are important SV features that help in Verification?

Another important homework for you is:
1. Search jobs in Verification & check which have SV mentioned in the skillsets - Every Verification job needs SV.

DAY 1

100 DAY VERIFICATION

i.) why SystemVerilog (SV) is used in Verification?

∴ SV has an oops Concept, which is mainly required in Verification.

- SystemVerilog provide support for gatelevel, RTL, Coverage, assertion & Constructs.
- SystemVerilog provides some additional Constructs for randomization implementation & oops techniques for improving the verification enviornment.

∴ So we can say that when design Complexity increses so does the requirement of better tools to design & verify it. & the SystemVerilog has ability to perform Constrained random Stimulation.

ii.) what is difference Between SV & UVM?

∴ SystemVerilog is a language that used to model hardware design & to verify design simulations.

∴ UVM is set of class libraries & a methodology recommended while building Testbench in SV.

∴ we can not Compare SystemVerilog and UVM as UVM is a methodology in which a rich set of functions are developed in its library using SV features make code-re usable and we structed.

iii.) which are more important SystemVerilog (SV) features that helps in Verification?

SystemVerilog provides important features Constrained-randomization, Coverage & assertions.

- Support for Constrained randomization, which can be used to generate a wide rage of test inputs. Similar Coverage which allows for the tracking & analysis of the functional Coverage of a design.

Summary Day ①

- The reason why we use Sy?
  - verification is that it allows structure to verify complex design, support randomization which will work automatic, concept of interface, where we can signal easily.
  - Another most important feature sv have over verilog is there is no race condition Between design and Testbench. which has in verilog so conflict so there is conflict in verilog.

- important feature of sv are:-
  - (a) Support oops Concept like inheritance, polymorphism, abstraction & encapsulation.
  - (b) Randomization with Constraint
  - (c) Keyword like logic, bit.
  - (d) Inface Concept.
  - (e) Code reusability. ∴ These make the verification process a lot easier.

- difference Between Sv and uvm:-
  - main difference Between Sv and uvm. Sv is Hardware Verification language and uvm is methodology, also called Suppenset of sv.
  - System Verilog (sv) can also be used to design but uvm can't.
  - Code reusability, scalability both of these achieved through uvm, not through sv.
  - uvm is base class library which helps to use feature of uvm which is not possible through sv.
  - writing Testbenches in sv using uvm increases reusability, interoperability, easy debugging, Configuration at top level, Synchronize etc.

Topic: Cache Mapping

Homework: Read about various mapping techniques in cache - Direct, Associative, Set associative. Understand every detail & how mapping is done.

After going through the concept, answer below questions:
1. What are pros & cons of every mapping technique?
2. Which mapping technique is most optimized?
3. I have a RAM of 1 MB & cache memory of 4 KB in a word addressable system, where 1 word is 4 bytes. Consider every frame consists of 4 words, which means every frame has 4*4 bytes = 16 bytes. How are we going to map the frames from RAM in cache memory for each mapping technique? (In case of set-associative consider 4-way)

Tips to Solve this numerical:
Fully Set Associative - any frame can be placed in any cache line, so no need to solve. Any block of RAM can be moved to any cache line.
Direct mapping - find out which frame nos. in RAM will be mapped to which cache lines.
Set Associative - first calculate how many bits are for tag, set & frame offset. Now, calculate which frames in RAM can be mapped to which set(it can go to any of the 4 ways in the set)

#100DAYSVERIFICATION

DAY : 2

Topic : CACHE MAPPING

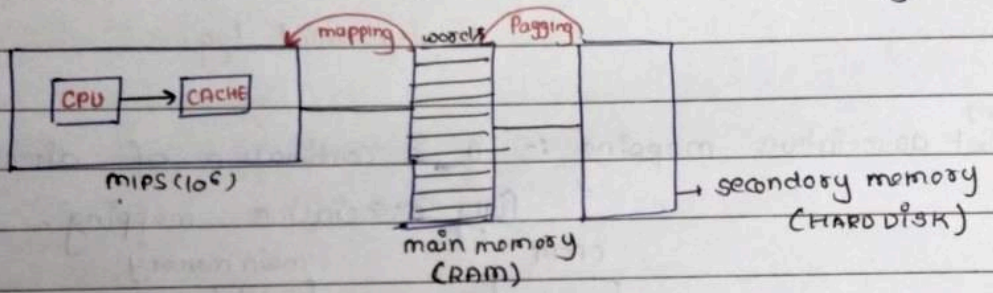
1) What is CACHE MAPPING? what are pros & cons of every mapping technique?

→ Cache as a mechanism to Speed up memory access. its High speed memory. OR cache memory is a small size memory in Betwee CPU and main memory.

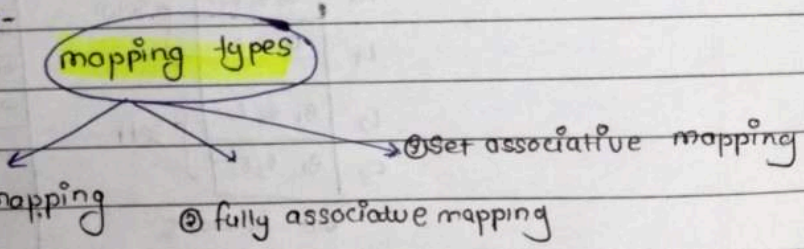
- CACHE MAPPING:-

Cache mapping is a technique By which content of main memory is brought in to the cache memor.

- OR It is a transformation of data from main memory to cache memory

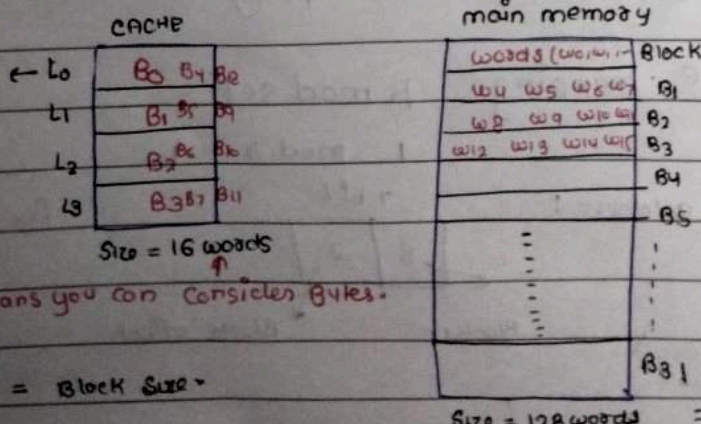


- CACHE mapping types :-



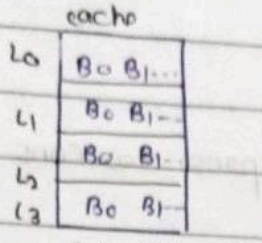
1) direct mapping :- for direct mapping $\frac{K}{\text{Block No.}} \bmod \underbrace{n}_{\text{No of lines}}$

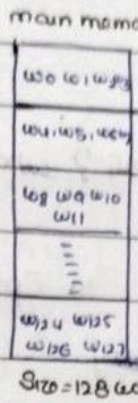
B_0 (0 mod 4)



Line Size = Block Size.

② Fully associative mapping :- Line size = Block size

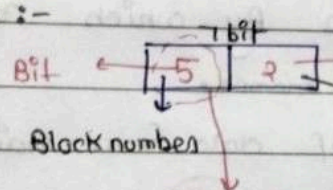




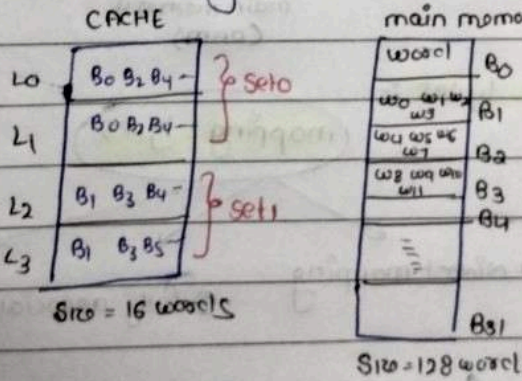
(Page is 0 to 127)

→ in case of fully associative mapping any block can come anywhere.

- physical address :-



③ K way Set associative mapping :- It is combination of direct mapping and fully associative mapping.



∴ K way = No. of lines / (No of set) ⇒ / K

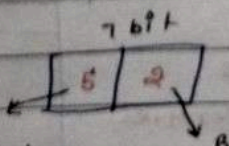
in direct mapping

K mod (n) — but for set mapping we take set

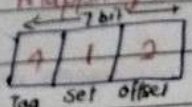
in associative mapping K mod set

∴ Physical address :

1 mod 2 = 1



for mapping :



i) Pross & Cons for every mapping :-

⇒ Direct mapping :

Pros	Cons
i) Searching time will be less & less expensive than associative mapping	i) It gives low performance because of the replacement for data-tag values

⇒ fully associative mapping

Pros	Cons
i) any block can come any location ii) number of conflict misses more so no of hit has increased.	i) no. of conflict misses is more ii) Comparision time have increased. then you understand got hit or miss.

⇒ Set Associative mapping :-

i) It gives better Performance than the direct and associative mapping	i) It is most expensive as with the increase in set size cost also increase.

ii) which mapping technique is most optimized ?

∴ set associative mapping is most optimized because it allows memory blocks to be mapped to multiple cache locations. its more flexibiliy in caching & can reduce the frequency of cache missee. its also provide faster assess times by holding more data in the cache.

SUMMARY : DAY (2)

⊙ There are 3 cache mapping Techniques :-

① direct mapping: each block of main memory is mapped to a specific location in the cache.

∴ PROS: its straight forward and efficient. data from main memory direcly copied to cache.

∴ Cons: lead to cache conflicts when data is read from multiple memories.

② Set Associative mapping: each cache location can store multipla Blocks of main memory. The cache is divided into sets. each set contains a fixed number of cache lines, each Block of main memory can be mappen to any cache line in any set.

∴ Pros: Reduces the ~~chance~~ chance of cache Conflicts.

∴ Cons: Complex to implement.

③ Fully Associative mapping: each Block of main memory can be placed in any location of cache.

∴ PROS: most flexibuly in copying data from memory and reduces cache Conflicts.

∴ Cons: Complex to manage and slower.

⊙ which mapping technique is most optimized?

- Set associative mapping is most optimized because it use both direct mapping & associative mapping techique. & becaus in cache consist of a number sets, each of which consist of numbes of Blocks
- Set associave mapping often Considered well-optimized as it provided a good balance Btw Complexity & Cache Conflict reduction.

DAY 3 - 100 DAYS VERIFICATION CHALLENGE

TOPIC: **CACHE REPLACEMENT ALGORITHMS**

DAY3 VERIFICATION CHALLENGE HOMEWORK: Understand Cache replacement algorithms for different mapping techniques. There are 3 major algorithms – LRU, MRU, FIFO. Analyze how replacement happens in each of these algos for different mapping techniques.

DAY3 VERIFICATION CHALLENGE:

1. What is the basic replacement criteria in all 3 algos?
2. Why does direct mapping not need any replacement algorithm?
3. I have a cache which can hold 8 blocks/frames from the RAM. Consider the cache is initially EMPTY. RAM has 128 blocks of 1 byte each in a byte addressable system. The processor requested blocks in following sequence: 127, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 0, 5, 11, 19, 8. Calculate the no. of cache misses for Fully Associative Cache Mapping in LRU, MRU & FIFO.

#100DaysVerification

Day-3

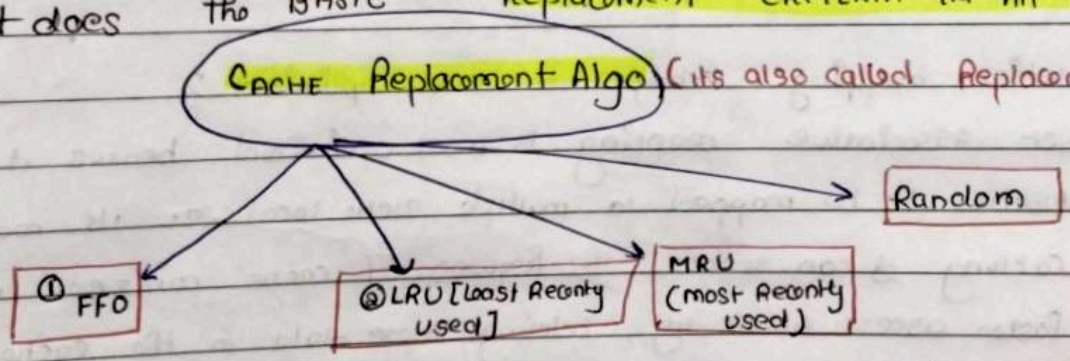
Topic : CACHE REPLACEMENT ALGORITHMS

→ Why we need for Replacement Algorithm ? [FOR DIRECT MAPPING]

∴ in case of direct mapping no need for a replacet algorithm.

- its because of the Block of the main memory would be able to map to a certain line of the cache only.
- Thus, The incoming (new) Block always happens to replace the Block that already exits, if any, in the certain line.
- in case of fully associative mapping The replacement algorithm is always required.
- Replacement algorithm suggests a Block that is to be replaced whenever all the cache line happen to be occupied.
- Similar set associative mapping also needs a certain type replacement algorithm.

Q) what does the BASIC Replacement CRITERIA IN All 3 Algos ?

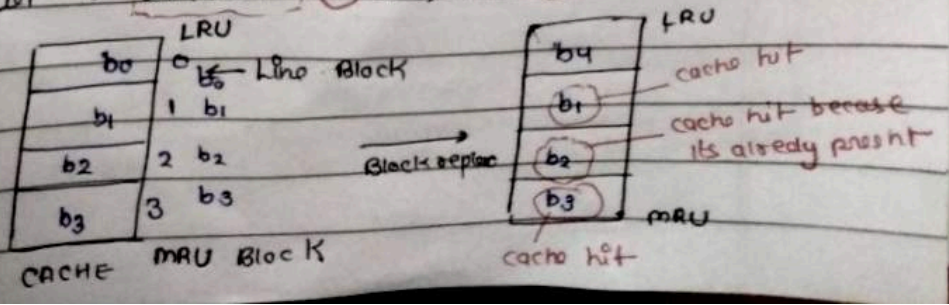


→ There are multiple way to replace cache, no need to replace cache but we need to replacing the data in cache.

◉ LRU [Least Replac Recently used] Cache Replacement Algorithm

- exploits Temporal Locality.
- Elict Least Recently referred Block.

Block Request : 0, 1, 2, 3, (4), 2, 3, 1, (5), 6 (cache is Full; here cache full (we need here replacement))



MRU [most Recently used] :-

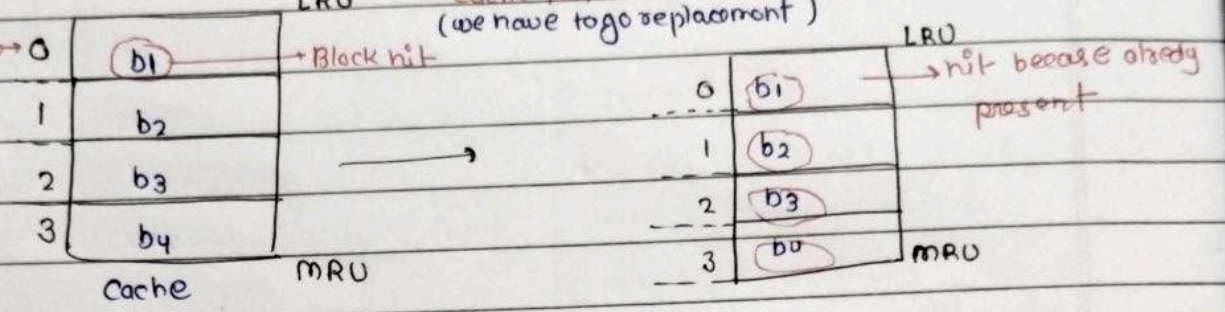
- Evicts most Recently accessed Block.
- Works well with cyclic patterns.

Block request : 1, 2, 3, 4, 5, 1, 2, 3, 4, 1, 2

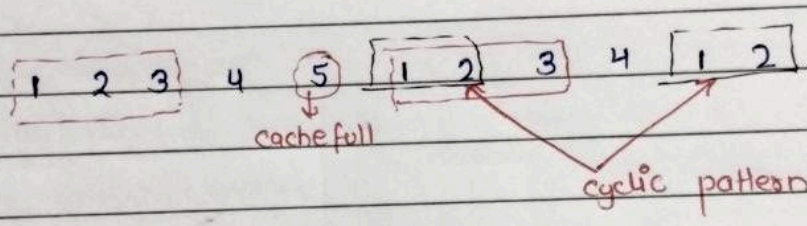
(hit (main memory already present))

cache again full

cache full (we have to go replacement)

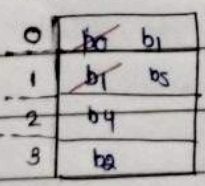


∴ So this is how the most recently used cache replacement policy works.



⊙ FIFO [FIRST IN FIRST OUT]

- Evicts Blocks from cache in their order of arrival.
- Cache Behave as a FIRST-IN-FIRST Out Queue.



$$\text{Miss Ratio} = \frac{\text{No. of misses}}{\text{Total No. of Request}} \times 100\%$$

$$\text{Hit Ratio} = (100 - \text{Miss Ratio})\%$$

⊙ Random Replacement :-

- Evicts any Block from cache at random.
- Access information is not needed.
- NOT implemented (used to be implemented in ARM architecture)

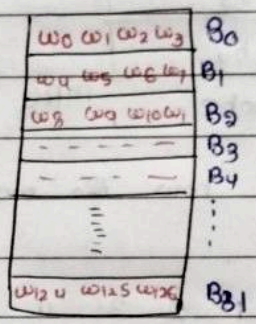
③ **REPLACEMENT ALGO BASED PROBLEM**

i have cache which can hold 8 Blocks / frames from the RAM. Consider the cache is initialy empty. RAM has 128 Blocks of 1 Byte each in a byte addressable system. The processor requested Blocks in following Sequence 128, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 10, 5, 11, 19, 8. Calculate the No of cache misses for fully associative Cache mapping in LRU, MRU & FIFO.

main memory = 128 Block

$= \frac{128}{4}$

$= 7$ Bit

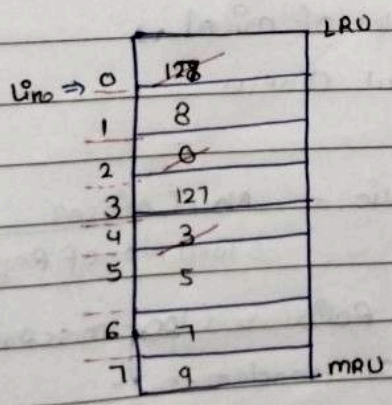


main memory
(RAM)
size = 128

∴ Block request : 128, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 0, 5, 11, 19, 8

cache is ff

:- LRU

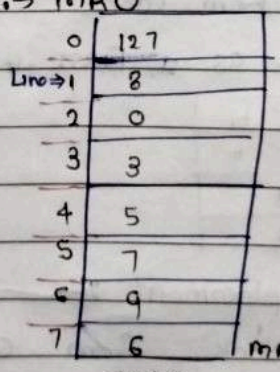


Cache

Total miss is 11.

No of HIT = 6

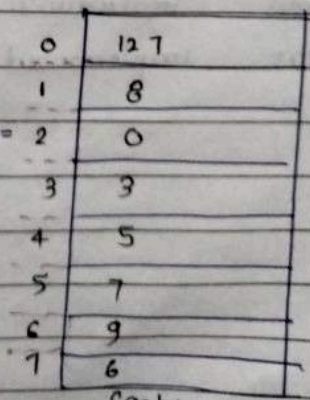
:⇒ MRU



Cache

⇒ Total cache miss = 10

LIFO :-



Cache

No of Miss = 11

Total hit = 5

③ **REPLACEMENT ALGO BASED PROBLEM**

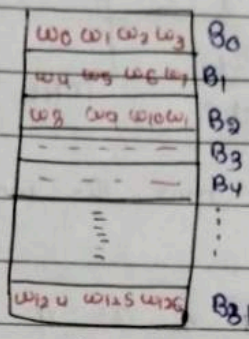
i have cache which can hold 8 Blocks / frames from the RAM. Consider the cache is initialy empty. RAM has 128 Blocks of 1 Byte each in a byte addressable system. The processor requested Blocks in following Sequence 127, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 0, 5, 11, 19, 8. Calculate the No of cache misses for fully associative Cache mapping in LRU, MRU & FIFO.

main memory = 128 Block

$= \frac{128}{4}$

Cache Line = 7 Bit

$\Rightarrow 8$



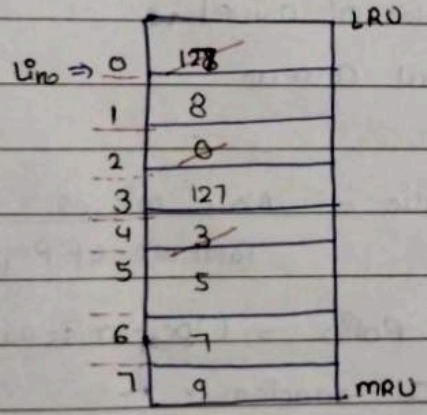
main memory (RAM)

Size = 128

∴ Block request : 127, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 0, 5, 11, 19, 8

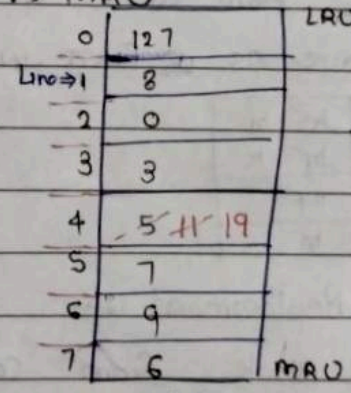
cache is ff

:- LRU



Cache

:⇒ MRU



Cache

FOR MRU :-

⇒ Total cache miss=10

M M M H M M M M
127, 8, 0, 127, 3, 5, 7, 9

M H H H M M M H
6, 3, 8, 0, 5, 11, 19, 8.

Total miss is 10.

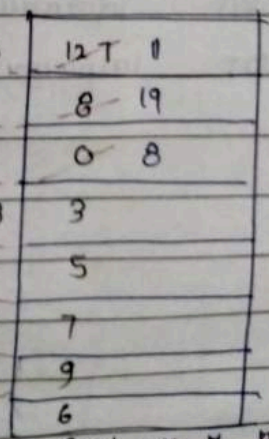
FOR LRU →

No of HIT = 6

M M M H M M M M M H
→ 127, 8, 0, 127, 3, 5, 7, 9, 6, 3

H H H M M H
8, 0, 5, 11, 19, 8.

LIFO :-



No of MISS=11

Total hit = 5

FOR FIFO : M M M H M M M M H H H M M M M Cache
127, 8, 0, 127, 3, 5, 7, 9, 6, 3, 8, 0, 5, 11, 19, 8.

NOTE BOOK

classmate
Date
Page

SUMMARY : DAY 3

① what is the Basic replacement criterics in all 3 algorithms ?

- The purpose of using replacement algorithms is to increase the number of hits in the cache memory.

i.> LRU: This algos means that the least recently used Block gets replaced By the new Block, its likely that a Block which has been accessed the least number of times will be accessed in the future.

ii.> MRU: This is the opposite of LRU, meaning the most recently used Block is replaced by a new Block, signifying that a Block that has Been used very recently is likely to be accessed further in the future.

iii.> FIFO: here we need to replace the Block that entered the RAM first. This is similar to an ATM queue where the person who arrived first for a transaction leaves first, or the first person is replaced by the new person.

② why does direct mapping not needed any replacement algorithm ?

→ direct mapping doesn't require replacement because here, the Block in the main memory is mapped to the same Block number in the cache main memory. for ex:- Block 0 (B0) of RAM is mapped to Block 0 in the cache, Block 1 of RAM is mapped to Block 1 in the cache, and so on. when we want to find a specific Block, only one search is required, if we find that Block, its a hit otherwise, it's a miss. means the disadvantage here is that the hit ratio is poor.

DAY 4 - 100 DAYS VERIFICATION CHALLENGE

TOPIC: **CACHE COHERENCE**

DAY4 VERIFICATION CHALLENGE HOMEWORK: Understand Cache coherency, cache coherency mechanisms & cache coherence protocols?

DAY4 VERIFICATION CHALLENGE:

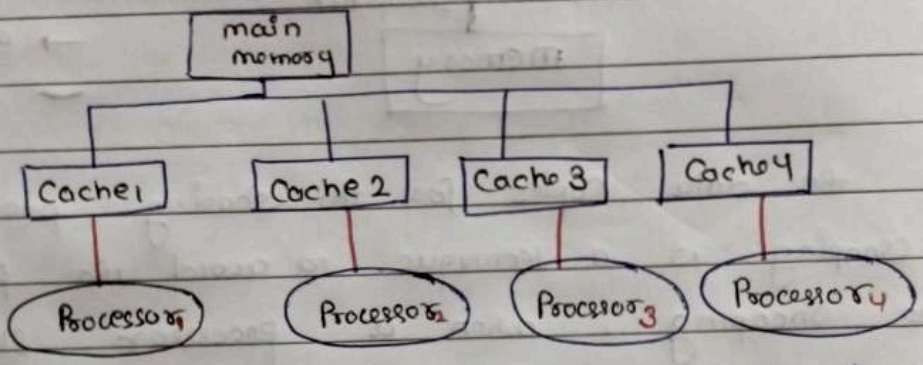
1. What is cache coherency problem?
2. Explain snooping mechanism?
3. What are cache coherency protocol? Explain below protocols:-
   - Write-invalidate
   - Write-update

#100DAYSVERIFICATION

DAY: 4

④ Topic : CACHE COHERENCE

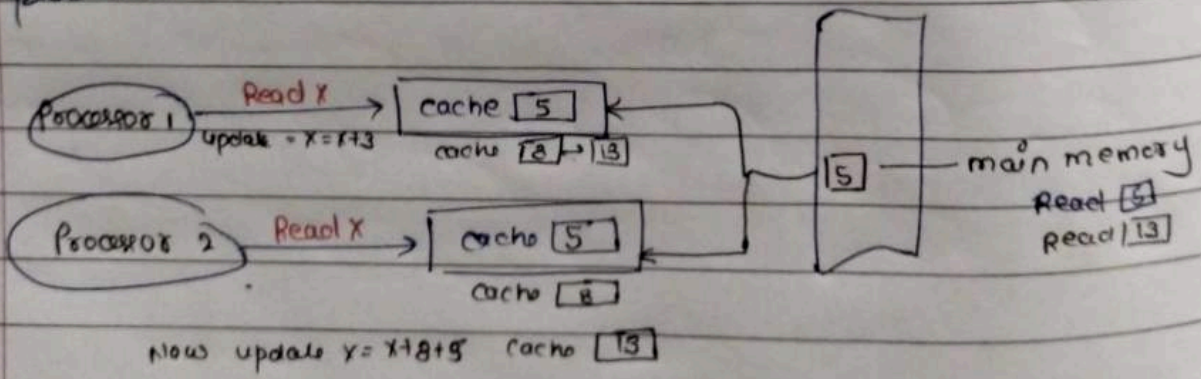
- UNDERSTAND CACHE Coherency :-
- for higher performance in a multiprocessor System, each processor usally have its own cache.
- in a multiprocessor System, data inconsistency may occur adjacent level or within the same level of memory hierarchy.
  eg :- The cache and the main memory may have inconsistent copies of the same object.



- we can take here n number of processor. like in my System have n processor and every processor have private cache.

1=> what is cache coherency problem ?

→ as we have multiple processors operates in parallel, and independently multiple caches may process different copies of the same memory Block, this create a cache coherency problem.

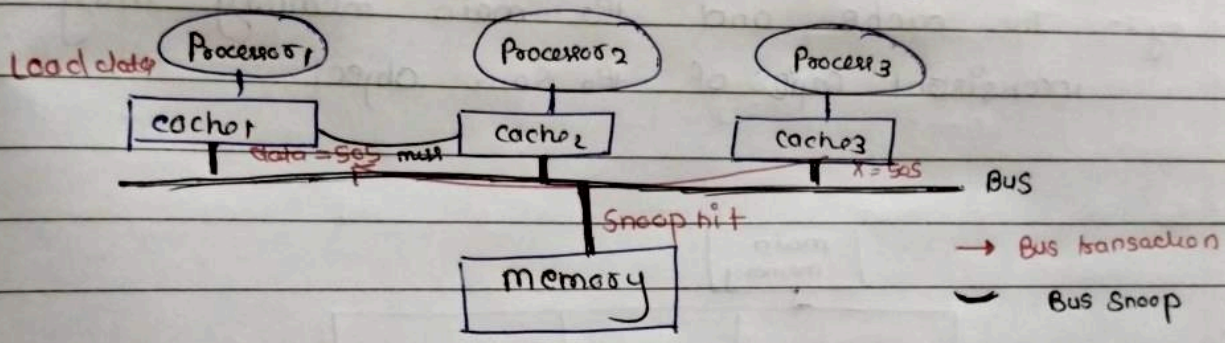


- cache coherence refer to the problem of keeping the data in those cache Consistance. The main problem is dealing with writes by a processor.

Sol'n ⑨ Explain Snooping mechanism?

∴ Snooping is a process where the individual caches monitor address line for access to memory locations that they cached. It is called a write invalidate protocol.

- When a write operation is observed to a location that a cache has a copy of and the cache controller invalidates its own copy of the snooped memory location.



- Invalidate the data copies for the sharing processor nodes.
- cache snoping is a technique to avoid the problem above.
- in cache snooping, when a processor stores a particular data in it cache, thereafter it monitors the access to the same data by any other processor(s)

- So no two processors works on the same data in a manner which can cause the data incoherence, which is nothing but different values for a single memory location.

③ what are cache coherency protocols ? Explains below protocols:

i) write-invalidate

ii) write-update.

→ **Cache coherency Protocols:-** There are various cache coherence protocol in multiprocessor system.

i) MSI Protocol:

- modified
- Shared
- invalid

ii) MOSI Protocol:

∴ This protocol is an extension of MSI Protocol. it adds the following state in MSI Protocol
- Owned.

iii) MESI Protocol: most widely used cache coherence protocol.

- modified
- Exclusive
- Shared
- invalid.

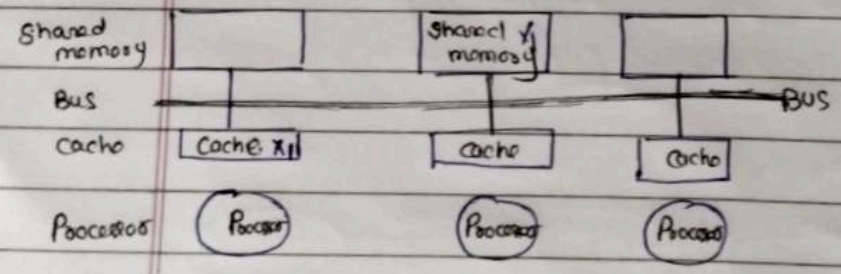
iv) MOESI Protocol: This is full cache coherence protocol that encompasses all of the possible states commonly used in other protocols.

- modified
- owned
- Exclusive
- Shared
- invalid

:- **Cache Coherency protocols:**

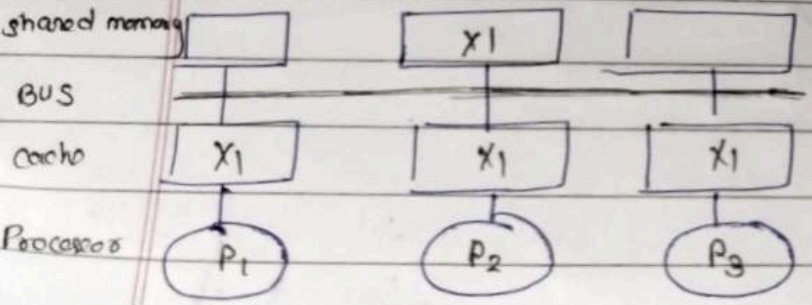
→ Snooping-based protocol | Bus-Based protocol
- write update
- write-invalidate

i) **write-invalidate:-** means when local cache copy is modified, write-invalidated policy. it remove invalidates all remote copies of cache (invalidated items are sometimes called "dirty"). means they should not be used



if processor P_1 modified (writes) in cache from X to X_1, then all other copies are invalidated.

ii) **write update:-** The new modified content X_1 be broadcast (upda) to all cache copies via bus.



- when a local cache copy is updated, write update policy broadcast a modified value of a data object to all other caches at the time of modification.