Multiprogram : Multiple programs are being run with the same CPU with shared memory

Multi-threading : A single task is divided into subtasks and is being run by different CPUs parallelly.

Multiprogram : Memory, CPU

Multi-thread : ?

Multiprogram Vs Multithread !

MSI protocol:

Cache coherence diagram:

Cache block state : M, S, I

M : Modified state
S : Share state
I : Invalid state / NP : Not Present

M ⇒ Core can modify

S ⇒ Core can only read. No write

I ⇒ If block is not in a cache, we can assume its in Cache in I/NP state

Events:

- GETS
- GETX
- UPGRADE
- PUTX
- REPLACE
- REPLACE_CLEAN

Appendix C

Should be there in main too?

Inclusive cache : A copy should exist at all levels. Better for managment
Exclusive cache : Not necessary to keep a copy at all levels. Better for space.

(g) Assembly lang code

lw: Load word 2 into 1 — add: Add 2,3 & store in 1.
sw: Store word 1 into 2

Mem→Reg `lw $1, 0($2)` → Load the word at $2 with 0 offset into $1

Mem→Reg `lw $2, 4($2)` → Load the word at $2 with 4 offset into $2

Reg→Reg `add $3, $1, $2` → Add the values in register $1 and $2, store in $3

Reg→Mem `sw $3, 16($2)` → Store the word in $3 in $2 with offset 16.

GETS (Get Shared):
- Cache State: Not in L1
- Action: Read, no modification
- Result: Block is brought into the cache in the Shared state

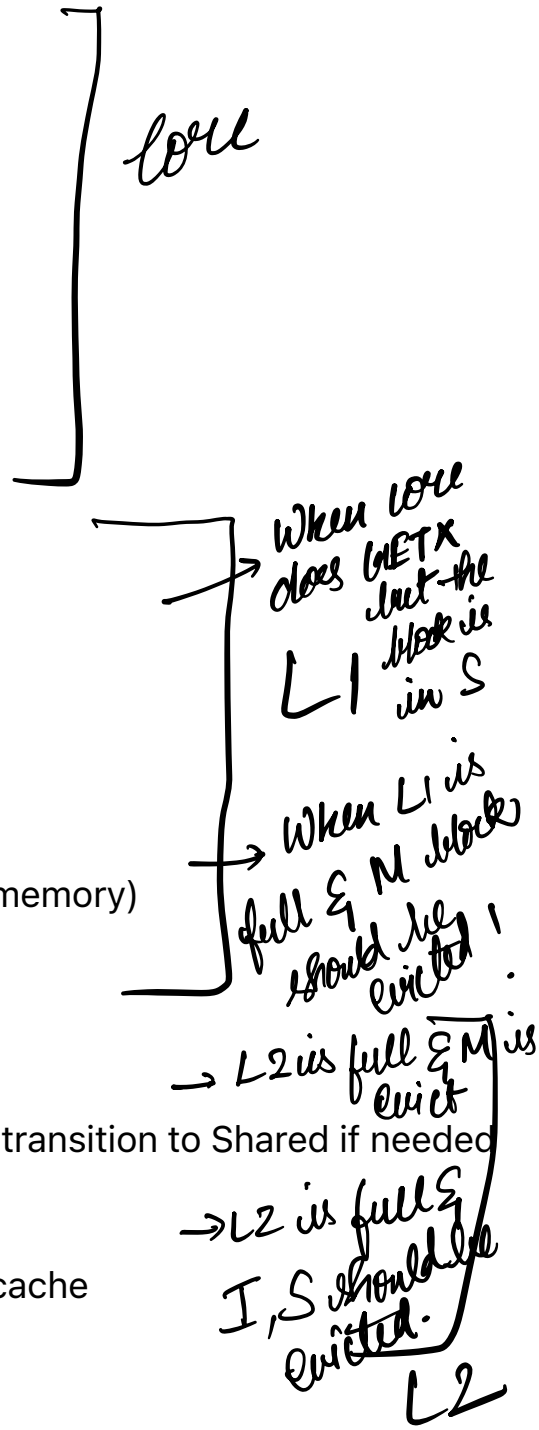
GETX (Get Exclusive):
- Cache State: Not in L1
- Action: Read, with intention to modify
- Result: Block transitions from Shared to Modified

UPGRADE:
- Cache State: Already in L1 (in Shared state)
- Action: Read, with intention to modify
- Result: Block transitions from Shared to Modified

PUTX (Put Exclusive):
- Cache State: Already in L1 (Modified state)
- Action: Write back the modified data to a lower level (L2 or memory)
- Result: Block transitions from Modified to Shared

REPLACE:
- Action: Evict a block in the Modified state from the cache
- Result: Write the modified block back to the lower level and transition to Shared if needed

REPLACE_CLEAN:
- Action: Evict a block in the Shared or Invalid state from the cache
- Result: No write-back required (since the block is clean)

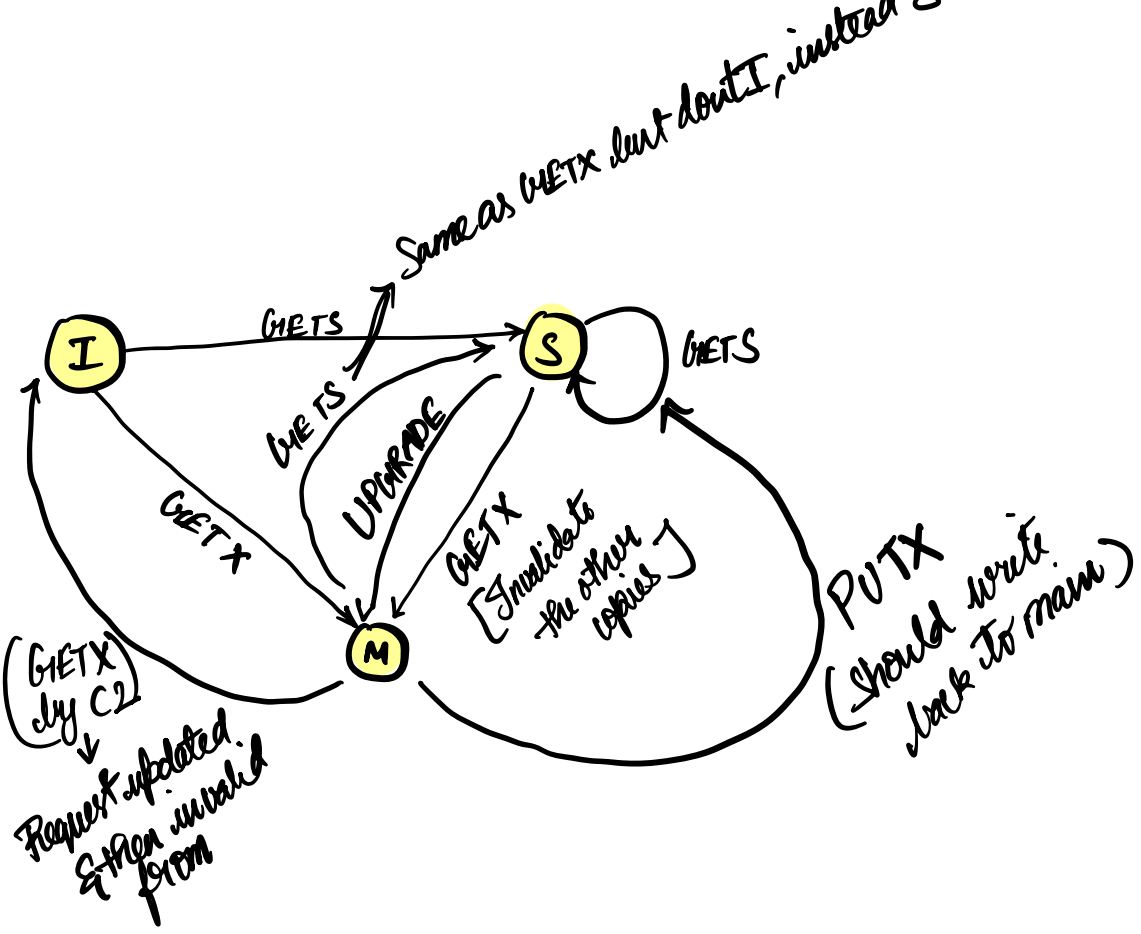


PutX :

→ Write back cache

→ Write throw cache

FSM?



Curr State	Event
I	GETS
I	GETX
I	Other
S	GETS
S	GETX
S	UPGRADE
S	PUTX
M	GETS
M	GETX
M	UPGRADE
M	PUTX

Curr State	Event
S	REPLACE
S	REPLACE_CLEAN
M	REPLACE
M	REPLACE_CLEAN

States : M, S, I.

Transient state : ST → Processing another request while in the process of a previous request.

→ Say C_1 is doing GETS for a block in I. It will take some time to fetch it.

→ In this time if C_2 is also doing GETS for the same block, the state is changed to ST.