CA HW 6

1) LSQ:-

LD/ST	Add Op.	ST Op.	Add. Val	Add. Cal	Mem. access
LD	2	-	abcd	3	4
ST	7	10	abde	8	commit
LD	5	-	abde	6	11
LD	3	-	abcd	4	9
ST	3	4	abde	4	commit
LD	1	-	abde	2	5
LD	4	-	abcd	5	9

2) Memory Access Time :-

Given:- L1 : 32KB : 1 cycle : 60 MPKI

L2 : 512KB : 5-cycle : 25 MPKI

L3 : 4 MB : 25-cycle : 15 MPKI

L4 : 32MB : 50-cycle : 3 MPKI

Memory : 250 cycles.

Execution time for:

① L1 - L2 - L3 - L4 - Memory

Let us assume that the program has 1000 instructions and since it has an IPC of 1., ideal execution time is 1000 cycles

Now for this hierarchy :-

$1000 + 60 \times 5 + 25 \times 25 + 15 \times 50 + 3 \times 250$

$= 1000 + 300 + 625 + 750 + 750 =$ **3425 cycles.**

② L1 - L2 - L3 - Memory

$1000 + 60 \times 5 + 25 \times 25 + 15 \times 250$

$= 1000 + 300 + 625 + 3750 =$ **5675 cycles**

③ L1 - L2 - L4 - Memory

$1000 + 60 \times 5 + 25 \times 50 + 3 \times 250$

$= 1000 + 300 + 1250 + 750 =$ **3300 cycles.**

3) Cache Organization

Given :- L3 cache = 40 MB address size = 48 b.

block size = 128 B

20-way set-associative

We know

~~Total cache size = #sets x 20x128B~~ [size of 1set = ~~# ways x block~~

Cache size = # sets x # ways x block size.

$\Rightarrow$ 40 MB = #sets x 20 x 128 B

$\therefore$ no. of sets = $\frac{40\,MB}{20 \times 128\,B} = \frac{40 \times 2^{20}}{20 \times 2^{7}}$

$= 2^{21-7} =$ **2^{14} sets**

Now index bits = $\log_2$ (#sets)

$= \log_2 (2^{14}) =$ **14 bits index**

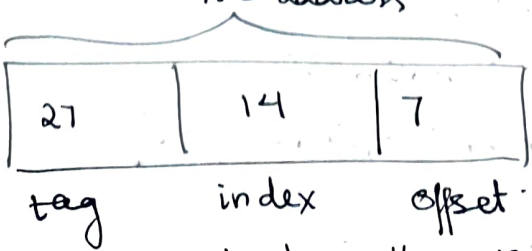
offset ~~tag~~ bits = $\log_2$ (block size)

$= \log_2 (2^{7}) =$ **7 bits offset**

Now tag bits = address size - (index + offset)

$= 48 - (14 + 7)$

$= 48 - 21 =$ **27 bits tag**



Tag array size = #sets x #ways x tag size

$= 2^{14} \times 20 \times 27 = 2^{14} \times 540 = 8847360\,b =$ **8640 Kb**

4) Cache Miss Rate

Case 1 - 2 sets and 2-way set associative

A, C, E → 0 set

B, D, F → 1 set.

Replacement Policy - LRU

(i) Access Pattern:

A	B	C	A	B	D	E	F	B	D	F	E	C	A	E	C	A
M	M	M	H	H	M	M	M	M	M	M	H	M	M	M	M	M

(ii) Hit rate = 3/17

A C E A	C F A C
B F D	D B F

Case 2 - Fully associative cache 1 set 4 ways

(i) Access Pattern:

A	B	C	A	B	D	E	F	B	D	F	E	C	A	E	C	A
M	M	M	H	H	M	M	M	H	H	H	H	M	M	H	H	H

(ii) Hit rate = 9/17

∴ We can see that hit rate has improved when we take a fully-associative cache.

A F	B C	C E	D A