INDIAN INSTITUTE OF ENGINEERING SCIENCE AND TECHNOLOGY, SHIBPUR

B TECH 5th SEMESTER MID-TERM EXAMINATION, OCTOBER 2021

COMPUTER ARCHITECHTURE AND ORGANIZATION [CS3103]

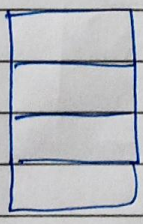
Name: Abhiroop Mukherjee
Enrolment No: 51051910 9
G-Suite ID: 510519109.abhiroop@students.iiests.ac.in
Date: 7th October 2021

Q1) A) → given 4 frames in cache
→ Block access sequence 0, 8, 0, 6, 8

→ Compare. direct, 2-way set associative & full associative in terms of no. of misses

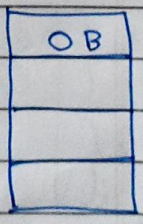
(I) direct

→ Initially

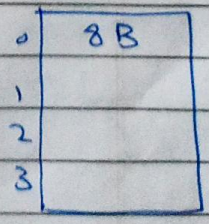


(empty cache)
(no. of frame = 4)

(i) → 0 block access.
→ compulsary miss
→ 0th block in 0%4 = 0th frame



(ii) → 8 block access.
→ compulsary miss
→ 8th block goes 8%4 = 0th frame.



(iii) 0^{th} block access

→ Conflict miss

→ 0^{th} block goes to $0\%4 = 0^{th}$ frame

0B	0
	1
	2
	3

(iv) 6^{th} block access

→ compulsary miss

→ 6^{th} block goes to $6\%4 = 2^{nd}$ frame

0B	0
	1
6B	2
	3

(v) 8^{th} block access

→ conflict miss

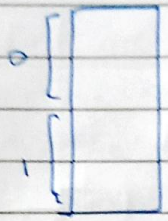
→ goes to 0^{th} frame

8B	0
	1
6B	2
	3

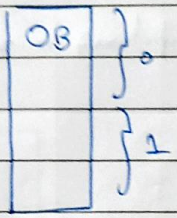
∴ Total misses = 5

II 2 way set associative

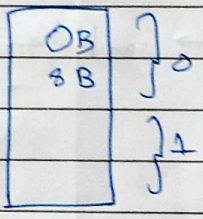
Intial



(i) 0^{th} block
→ compulsary miss
→ goes to $0\%2 = 0^{th}$ set

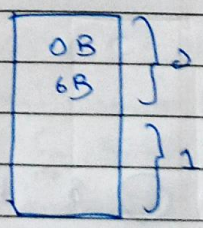


(ii) 8^{th} block
→ compulsary miss
→ goes to $8\%2 = 0^{th}$ set

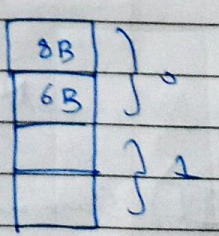


(iii) 0^{th} block
→ hit

(iv) 6^{th} block
→ compulsary miss
→ goes to $6\%2 = 0^{th}$ set
→ assuming oldest accessed block goes first.
→ 8^{th} block gone



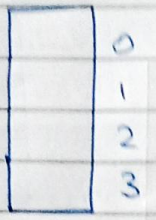
(v) 8^{th} block
→ conflict / capacity miss
→ assuming oldest accessed block goes first
→ 0^{th} block gone



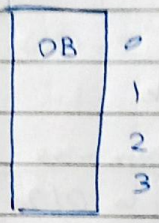
∴ Total miss = 4

III) Fully associative

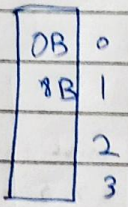
(i) Initial



(i) 0th block
→ compulsory miss

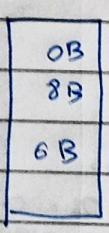


(ii) 3rd block
→ compulsory miss



(iii) 0th block
→ hit

(iv) 6th block
→ compulsory miss



(v) 8th block
→ hit

No. of miss = 3

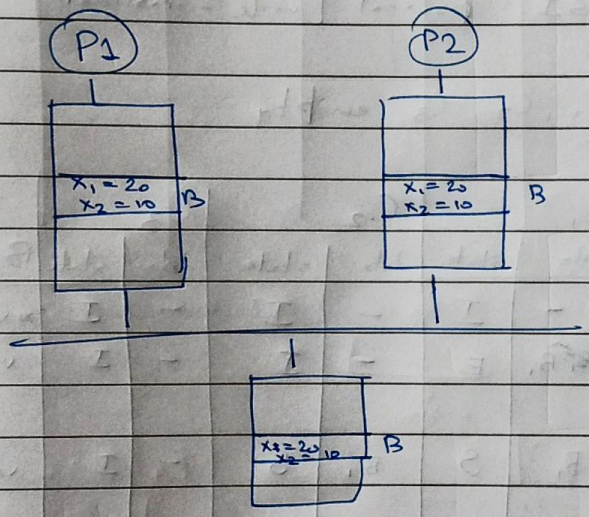
IV) Summary : in terms of misses in this case

Full associative < 2-way set associative < direct

(2)B) 'False sharing miss normally occurs in a system realizing the large block size & write-invalidate scheme for cache coherence'

→ This happens because in a large block size frame, ~~the~~ a modification to one value in block invalidates whole block for other processes. This leads to misses for other values in block, even if they are unmodified

Eg



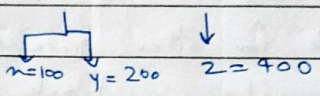
activity	C_1	C_2	remark
① initial	$X_1 = 20$ $X_2 = 10$	$X_1 = 20$ $X_2 = 10$	
② P_1 write X_1 → invalidate C_2	$X_1 = 30$ $X_2 = 10$	—	
③ P_2 read X_2 → miss (false)	$X_1 = 30$ $X_2 = 10$	$X_1 = 30$ $X_2 = 10$	got read miss for X_2 even when X_2 not modified
④ P_2 write X_2	—	$X_1 = 30$ $X_2 = 20$	
⑤ P_1 ~~write~~ ~~read~~ X_1 → write miss	$X_1 = 40$ $X_2 = 20$	—	got write miss for X_1, even when X_1 not modified

Q.) c) → three processor P_1, P_2, P_3

with caches C_1, C_2, C_3

→ caches can accomodate one block at a time.

→ MM stores two block B_1 & B_2



→ snoopy MESI protocol [write-invalidate]

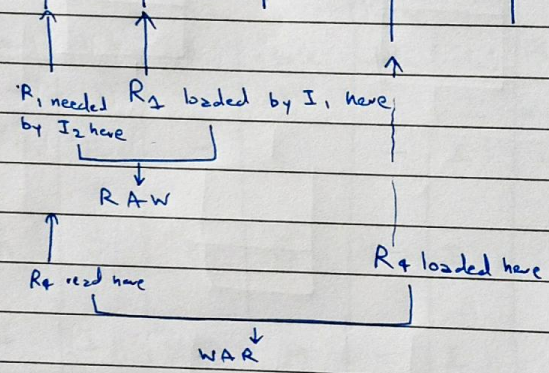
→ initially C_1, C_2, C_3 empty

Activity	C_1		C_2		C_3		MM	
	data	state	data	state	data	state	B_1	B_2
⓪ initial	–	I	–	I	–	I	valid	valid
① P_1 read x → read miss	B_1	E	–	I	–	I	valid	valid
② P_2 read X → read miss	B_1	S	B_1	S	–	I	valid	valid
③ P2 update y=y+500 → write hit → invalidate	–	I	B_1 x=100 y=700	M	–	I	invalid	valid
④ P_3 read X → read miss → P_2 give X to MM & P_3	–	I	B_1	S	B_1	S	valid x=100 y=700	valid z=400
⑤ P_1 updates x=x-200 → read miss → then write-invalidate	B_1 x=-100 y=700	M	–	I	–	I	invalid	valid
⑥ P_3 read y → read miss → P_1 gives B_1 to MM & P_3	B_1	S	–	I	B_1	S	valid x=-100 y=700	valid

Activity	C_1		C_2		C_3		MM	
	data	state	data	state	data	state	B_1	B_2
	B_1	S	–	I	B_1	S	valid x=-100 y=700	valid z=400
P_2 update y=100 → read miss → then write-invali	–	I	B_1 x=-100 y=100	M	–	I	invalid	valid z=400
P_1 read y → read miss → P_2 give B_1 to MM & P_1	B_1	S	B_1	S	–	I	valid x=-100 y=100	valid z=400

(Q2) a) 5 state pipe → IF, ID/OF, EX, MEM, WB

clock	1	2	3	4	5	6	7	8	9	10
R1←M(R2)	IF_1	ID/OF_1	EX_1	MEM_1	WB_1					
MR3←R1*R4		IF_2	ID_2	EX_2	MEM_2	WB_2				
R4←R1+R5			IF_3	ID_3	EX_3	MEM_3	WB_3			
R6←R1-R7				IF_4	ID_4	EX_4	MEM_4	WB_4		



2) data dependencies

(i) I_1 I_2 → RAW

(ii) I_2 I_3 → WAR

b) If no data forwarding implemented, there will be data hazard

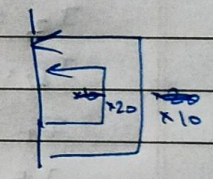
I_2 reads R_1 before R_1 set by I_1

c) possible solution with stall & data forwarding

	1	2	3	4	5	6	7	8
$R1 \leftarrow M[R2]$	IF$_1$	ID$_1$	EX$_1$	MEM$_1$	WB$_1$			
Stall				data forward				
$M_3 \leftarrow R_1 * R_4$			IF$_2$	ID$_2$	EX$_2$	MEM$_2$	WB$_2$	
$R_4 \leftarrow R_1 \vee R_5$				IF$_3$	ID$_3$	EX$_3$	MEM$_3$	WB$_3$
$R_5 \leftarrow R_1 \cdot R_7$					IF$_4$	ID$_4$	EX$_4$	

R$_1$ load R$_1$ read

2) B) outer loop 10, inner loop 20 time



a) always branch taken

outer loop → 10 time right, 1 wrong.
inner loop → 20 time right, 1 wrong.

$$\text{Total} = (20+1)\times 10 + 1 + 10 = 221$$

Correct = 210

$$\text{accuracy} = \frac{210}{221} = 95\%.$$

b) 1 bit history

→ for all inner loop, branch taken, except last
→ pred = 'not taken' after 20 iter inner loop
→ the mispridict pred = taken again
→ So in 9 iter of outer loop → wrong = 2 × 9 = 18
→ in last iter → 20 correct, 1 incorrect, 1 correct
no. of wrong = 19

$$\text{Accuracy} = \frac{221-9}{221} = 91\%$$

2) a) given original code.

I_1 : load R_1, X
I_2 : decrement R_2, 1
I_3 : Br. Zero R_2, I_5
I_4 : Add R_3, R_4
I_5 : Sub R_5, R_6

a) before branch prediction.

I_1 : load R_1, X
I_3 : Br. Zero R_2, I_5 } swapped.
I_2 : decrement R_2, R_1
I_4 : Add R_3, R_4
I_5 : Sub R_5, R_6

b) Target address

I_1 : load R_1, X
I_2 : decrement R_2, 1
I_3 : Br. Zero R_2, I_5
I_5 : Sub R_5, R_6
I_4 : Add R_3, R_4

→ for predict taken scheme
→ both (a) & (b) perform equally well