

Facendo riferimento al sistema S, si completi la mappa delle memoria con i due vettori x, y e la variabile z. Calcolare l'indirizzo del primo e ultimo elemento dell'array x considerato da ognuno dei cores. (Punti.3)

Si completi la mappa delle memoria con i due vettori x, y e la variabile z.					
BA[31..0]	Byte3	Byte2	Byte1	Byte0	BA[31..0]
0x010C 0003	z	z	z	z	0x010C 0000
0x010B 000F	y[3]	y[3]	y[3]	y[3]	0x010B 000C
0x010B 000B	y[2]	y[2]	y[2]	y[2]	0x010B 0008
0x010B 0007	y[1]	y[1]	y[1]	y[1]	0x010B 0004
0x010B 0003	y[0]	y[0]	y[0]	y[0]	0x010B 0000
0x010A 3FFF	x[4095]	x[4095]	x[4095]	x[4095]	0x010A 3FFC
...
0x010A 0003	x[0]	x[0]	x[0]	x[0]	0x010A 0000
	MSB			LSB	
COMMENTO: Considerando che un singolo valore float in singola precisione richiede 4 bytes per essere memorizzato, si ha che i vettori x e y e la variabile z hanno le seguenti dimensioni in memoria: x) 4096 elementi * 4 bytes = 16KiB (0x4000 bytes) y) 4 elementi * 4 bytes = 16 bytes (0x10 bytes) z) 1 elemento * 4 bytes = 4 bytes (0x4 bytes)					

Si compilino i campi in arancione con le risposte	
Nome	
Cognome	
Matricola	

Calcolare l'indirizzo del primo e ultimo elemento dell'array x considerato da ognuno dei cores. (Punti.3)									
		Cores							
		0		1		2		3	
		Inizio	Fine	Inizio	Fine	Inizio	Fine	Inizio	Fine
Array	Indice	0	1023	1024	2047	2048	3071	3072	4095
x	Indirizzo	0x010A 0000	0x010A 0FFC	0x010A 1000	0x010A 1FFFC	0x010A 2000	0x010A 2FFC	0x010A 3000	0x010A 3FFC
Array	Indice	0	3	0	3	0	3	0	3
tmp*	Indirizzo	0x0000 0000	0x0000 0003	0x0000 0000	0x0000 0003	0x0000 0000	0x0000 0003	0x0000 0000	0x0000 0003
*esempio di calcolo indirizzi per un array (tmp) di interi a 8 bit a indirizzo 0x0000 0000 (char tmp[4];) allocato in ogni core.									
COMMENTO: Ognuno dei 4 cores processa una porzione di array pari a un quarto degli elementi totali, quindi ogni core esegue il blocco di codice B0 su 4096 elementi / 4 cores = 1024 elementi. La dimensione di memoria per ogni porzione su cui un core lavora e' pari a 1024 elementi * 4 bytes = 4096 bytes (0x1000 bytes).									

2.b) Quali sono gli indici di set e linea, e i tag associati agli array x, y e z? Quante linee di cache occuperanno del loro insieme? Possono le 4 porzioni dell'array x essere contenute interamente nelle cache dei 4 cores?

Quali sono gli indici di set e linea, e i tag associati agli array x, y e z?

BA[31..0]	Contenuto Blocco di Cache			BA[31..0]		TAG	Set_id/indice	Numero progressivo delle linee/blocchi di cache occupati da x,y,z
Indirizzo primo elemento della linea/ blocco di cache	Da		A	Indirizzo ultimo elemento della linea/ blocco di cache		BA[?..?]	BA[?..?]	
	Si completi inserendo le variabili agli estremi del blocco di cache (Byte più significativo e meno significativo).					Si completino sotto i "?" identificando i bit utilizzati per tag e indice.		
						BA[?..?]	BA[?..?]	
0x010A 0000	x[0] LSB		x[7] MSB	0x010A 001F		0x010A0	0b00000000	1
0x010A 0020	x[8] LSB		x[15] MSB	0x010A 003F		0x010A0	0b00000001	2
...
0x010A 0FE0	x[1016] LSB		x[1023] MSB	0x010A 0FFF		0x010A0	0b11111111	128
0x010B 0000	y[0] LSB		-	0x010B 001F		0x010B0	0b00000000	129
0x010C 0000	z LSB		-	0x010C 001F		0x010C0	0b00000000	130

COMMENTO: Ogni core possiede una cache L1 privata della dimensione di 8KiB e linee da 32 bytes. Il numero totale di linee in cache vale $8 \text{ KiB} / 32 \text{ bytes} = 256$ linee, ma siccome la cache possiede due vie, ogni via ha $256 \text{ linee} / 2 = 128$ linee. Dalla struttura della cache si calcolano il numero di bits necessari a rappresentare TAG, index e bytes-in-line:

bytes-in-line) $32 \text{ bytes} = 5 \text{ bits}$

index) $128 \text{ linee} = 7 \text{ bits}$

TAG) $32 \text{ bits} - 5 \text{ bits} - 7 \text{ bits} = 20 \text{ bits}$

Quante linee di cache occuperanno del loro insieme? Possono le 4 porzioni dell'array x essere contenute interamente nelle cache dei 4 cores?

Ognuna delle quattro porzioni in cui si divide il vettore x si compone di $1024 \text{ elementi} * 4 \text{ bytes} = 4 \text{ KiB}$. Siccome ognuna delle due vie che compongono la cache puo' memorizzare fino a 4 KiB di dati, si puo' concludere che ogni porzione del vettore x e' contenuto interamente nella cache del core che lo processa.

2.c) Si consideri la dinamica della cache nel calcolo di $y[0]$ - blocco B0 - svolto solo dal core0 - disegnando lo stato MESI, il contenuto della cache e il valore del bit LRU dopo ogni operazione di accesso alla memoria (Load e Store) **per la prima e l'ultima iterazione del loop**. Si indichi il numero di miss, il numero di accessi, si calcoli la miss rate. Quant'è l'arithmetic intensity associata al blocco di codice B0?

Stato MESI	Iterazione:	0	Miss	1		Hit	7
	Istruzione:	flw f1, 0(x10)					
	Via0			LRU	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x010A0	x[0]	E	1			I
...			I	-			I
			I	-			I
Stato MESI	Iterazione:	0	Miss	128		Hit	896
	Istruzione:	flw f1, 0(x10)					
	Via0			LRU	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x010A0	x[0]	E	1			I
...			I
127	0x010A0	x[1016]	E	1			I
Stato MESI	Iterazione:	?	Miss	129		Hit	896
	Istruzione:	fsw f0, 0(x11)					
	Via0			LRU	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x010A0	x[0]	E	0	0x010B0	y[0]	M
...			I
127	0x010A0	x[1016]	E	1			I

Si indichi il numero di accessi, il numero di miss e si calcoli la miss rate. Quant'è l'arithmetic intensity associata al blocco di codice B0?

Ogni core esegue 1024 accessi in memoria in lettura e un singolo accesso in scrittura (fsw alla fine del blocco di codice B0). Se si assume la cache L1 di ogni core vuota all'inizio del loop, allora il miss rate è pari a 1 MISS ogni 8 accessi. La store (fsw) alla fine del loop è una miss, in quanto il vettore y viene utilizzato per la prima volta alla fine del loop. Per via della politica di write-allocate, il vettore y verrà caricato in cache nella prima linea della via1.

Il MISS rate della cache può essere approssimato a $1/8 \sim 12.5\%$.

Se nella MISS rate includiamo la MISS dovuta alla fsw ed alla politica di write-allocate, la MISS rate è pari a $129 / (129 + 896) \sim 12.6\%$.

L'arithmetic intensity del blocco di codice B0 è pari a 0.25. Infatti, per ogni elemento floating-point letto (4 bytes) si esegue una operazione di fadd.s.

2.d) Si determini il numero di cicli di WB nonché lo stato MESI, il contenuto e il valore del bit LRU della cache al termine dell'esecuzione parallela del blocco B0. Al fine della risoluzione del quesito si assuma che l'ultima istruzione del blocco parallelo (lsw) sia eseguita da tutti i processori con il seguente ordine: Core-0, Core-1, Core-2, Core-3. Si riporti lo stato delle prime due ed ultime due linee di cache. Se non incluse si riportino le eventuali linee di cache che contengono gli elementi dell'array y.

Stato MESI	Core0							Core1							Core2							Core3						
	Via0			LRU	Via1			Via0			LRU	Via1			Via0			LRU	Via1			Via0			LRU	Via1		
	set_id	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI	TAG	Data	MESI
0	0x010A0	x[0]	E	0	0x010B0	y[0]	I	0x010A1	x[1024]	E	0	0x010B0	y[0]	I	0x010A2	x[2048]	E	0	0x010B0	y[0]	I	0x010A3	x[3072]	E	0	0x010B0	y[0]	M
1	0x010A0	x[8]	E	1			I	0x010A1	x[1032]	E	1			I	0x010A2	x[2056]	E	1			I	0x010A3	x[3080]	E	1			I
...			I			I			I			I
126	0x010A0	x[1008]	E	1			I	0x010A1	x[2032]	E	1			I	0x010A2	x[3056]	E	1			I	0x010A3	x[4080]	E	1			I
127	0x010A0	x[1016]	E	1			I	0x010A1	x[2040]	E	1			I	0x010A2	x[3064]	E	1			I	0x010A3	x[4088]	E	1			I

I cicli totali di WB sono 3.

2.e) Come cambierebbe lo stato della cache se la politica di scrittura in caso di miss fosse Write Around?

Se gli eventi di cache miss in scrittura fossero gestiti con una politica di write-around, la scrittura presente alla fine del blocco di codice B0 (fsw) non scatenerrebbe un accesso in memoria da parte di ognuno dei cache controller per portare in cache il vettore y. Al contrario, ogni core accedrebbe alla gerarchia superiore di memoria per scrivere direttamente il contenuto della somma parziale appena calcolata. Quindi, non sarebbe presente nella via 1 (Set-Id 0) il TAG presente nella via 1 (Set-Id 0) il TAG (0x010B0).

2.f) Si mostri lo stato della cache per il core 3 alla fine dell'esecuzione della porzione seriale del blocco di codice. Si riporti lo stato delle prime due ed ultime due linee di cache. Se non incluse si riportino le eventuali linee di cache che contengono gli elementi dell'array y e della variabile z.

Stato MESI	Core3						
	Via0			LRU	Via1		
	set_id	TAG	Data		MESI	TAG	Data
0	0x010C0	z	M	1	0x010B0	y[0]	M
1	0x010A3	x[3080]	E	1			I
...
126	0x010A3	x[4080]	E	1			I
127	0x010A3	x[4088]	E	1			I