2.a) Facendo riferimento al sistema S, si completi la mappa delle memoria con i tre vettori x, y e z. Calcolare l'indirizzo del primo e ultimo elemento degli array x, y e z considerato da ognuno dei cores.

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

Si completi la mappa delle memoria con i tre vettori x, y e z.

BA[31..0]	Byte3	Byte2	Byte1	Byte0	BA[31..0]
0x0010 3FFF	z[511]	z[511]	z[511]	z[511]	0x0010 3FFC
0x0010 3FFB	z[511]	z[511]	z[511]	z[511]	0x0010 3FF8
…	…	…	…	…	…
0x0010 3007	z[0]	z[0]	z[0]	z[0]	0x0010 3004
0x0010 3003	z[0]	z[0]	z[0]	z[0]	0x0010 3000
0x0010 2FFF	y[511]	y[511]	y[511]	y[511]	0x0010 2FFC
0x0010 2FFB	y[511]	y[511]	y[511]	y[511]	0x0010 2FF8
…	…	…	…	…	…
0x0010 2007	y[0]	y[0]	y[0]	y[0]	0x0010 2004
0x0010 2003	y[0]	y[0]	y[0]	y[0]	0x0010 2000
0x0010 1FFF	x[511]	x[511]	x[511]	x[511]	0x0010 1FFC
0x0010 1FFB	x[511]	x[511]	x[511]	x[511]	0x0010 1FF8
…	…	…	…	…	…
0x0010 1007	x[0]	x[0]	x[0]	x[0]	0x0010 1004
0x0010 1003	x[0]	x[0]	x[0]	x[0]	0x0010 1000
	MSB			LSB	

Calcolare l'indirizzo del primo e ultimo elemento dell'array x,y e z considerato da ognuno dei cores.

		Cores			
		0		1	
		Inizio	Fine	Inizio	Fine
Array	**Indice**	0	255	256	511
x	**Indirizzo**	0x0010 1000	0x0010 17FF	0x0010 1800	0x0010 1FFF
Array	**Indice**	0	255	256	511
y	**Indirizzo**	0x0010 2000	0x0010 27FF	0x0010 2800	0x0010 2FFF
Array	**Indice**	0	255	256	511
z	**Indirizzo**	0x0010 3000	0x0010 37FF	0x0010 3800	0x0010 3FFF
Array	**Indice**	0	3	0	3
tmp	**Indirizzo**	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.

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 porzioni dell'array x,y e z associate al core 0 essere contenute interamente nelle sua cache L1 privata?

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[?..?]	
0x0010 1000	x[0] LSB		x[7] LSB	0x0010 1038		0x0 0202	0b00000	*1*
…	…		…	…		…	…	…
0x0010 1FC0	x[504] LSB		x[511] LSB	0x0010 1FF8		0x0 0203	0b11111	*63*
0x0010 2000	y[0] LSB		y[7] MSB	0x0010 2038		0x0 0204	0b00000	*64*
…	…		…	…			…	
0x0010 2FC0	y[504] LSB		y[511] LSB	0x0010 2FF8		0x0 0205	0b11111	*127*
0x0010 3000	z[0] LSB		z[7] LSB	0x0010 3038		0x0 0206	0b00000	*128*
…	…		…				…	
0x0010 3FC0	z[504] LSB		z[511] LSB	0x0010 3FF8		0x0 0207	0b11111	*192*

COMMENTO: Ogni core possiede una cache L1 privata della dimensione di 4KiB e linee da 64 bytes. Il numero totale di linee in cache vale 4 KiB / 64 bytes = 64 linee, ma siccome la cache possiede due vie, ogni via ha 64 linee / 2 = 32 linee. Dalla struttura della cache si calcolano il numero di bits necessari a rappresentare TAG, index e bytes-in-line:

bytes-in-line) 64 bytes = 6 bits

index) 32 linee = 5 bits

TAG) 32 bits - 5 bits - 6 bits = 21 bits

Quante linee di cache occuperanno del loro insieme? Possono le porzioni dell'array x,y e z associate al core 0 essere contenute contemporaneamente nelle sua cache L1 privata?

Ognuna delle porzioni in cui si divide il vettore x,y,z si compone di 256 elementi * 8 bytes = 2 KiB. Visto che la cache ha solo due vie e le porzioni di vettori condividono gli stessi set_id solo due dei vettori possono stare in cache contemporaneamente.

2.c) Si consideri la dinamica della cache nel calcolo dell'elemento z[0] svolto 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).

Stato MESI	**Iterazione:**	**0**	**Miss**	**1**		**Hit**	**0**
	Istruzione:			**fld f0, 0(x10)**			
	Via0			**LRU**	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x0 0202	x[0]	E	1			I
...			I	-			I
			I	-			I

Stato MESI	**Iterazione:**	**0**	**Miss**	**2**		**Hit**	**0**
	Istruzione:			**fld f1, 0(x11)**			
	Via0			**LRU**	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x0 0202	x[0]	E	0	0x0 0204	y[0]	E
...			I	-			I
			I	-			I

Stato MESI	**Iterazione:**	**?**	**Miss**	**3**		**Hit**	**0**
	Istruzione:			**fsd f0, 0(x12)**			
	Via0			**LRU**	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x0 0206	z[0]	M	1	0x0 0204	y[0]	E
...			I	-			I
			I	-			I

2.d) Si mostri lo stato MESI, il contenuto e il valore del bit LRU della cache per il solo core 0, ignorando i calcoli svolti dal core1. Si riporti lo stato della prima e dell'ultima linee di cache per il core.
Si indichi il numero di miss, il numero di accessi, si calcoli la miss rate e si calcoli il numero di cicli di WB.

Stato MESI	**Via 0**			**LRU**	**Via1**		
set_id	TAG	Data	MESI		TAG	Data	MESI
0	0x0 0204	y[7]	E	0	0x0 0206	z[7]	M
…	..	…	…	…	…	…	…
31	0x0 0204	y[255]	E	0	0x0 0206	z[255]	M

Si indichi il numero di miss, il numero di accessi, si calcoli la miss rate e si calcoli il numero di cicli di WB.

Ogni core esegue 512 accessi in memoria in lettura e 256 accessi in scrittura. Se si assume la cache L1 di ogni core vuota all'inizio del loop, allora il miss rate e' pari a 1 MISS per ogni accesso.

Il MISS rate della cache puo' essere approssimato a 100%.
Il numero di cicli di WB è pari a 256-32 = 224

2.e) Come cambierebbero i cicli di WB e la miss rate 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, non si avrebbe nessun ciclo di WB e solo miss compulsorie in lettura.

Ciò darebbe origine a una miss rate considerando solo gli accessi in lettura pari a 1/8.

2.f) Si mostri lo stato MESI, il contenuto e il valore del bit LRU della cache per entrambi i core al fine dell'esecuzione del codice sul core1. Facendo l'ipotesi che il core1 esegua dopo il termine dell'esecuzione sul core0. Si riporti lo stato della prima e dell'ultima linea di cache.

Stato MESI	Core0							Core1						
	Via0			LRU	Via1			Via0			LRU	Via1		
set_id	TAG	Data	MESI		TAG	Data	MESI	TAG	Data	MESI		TAG	Data	MESI
0	0x0 0204	y[7]	E	0	0x0 0206	z[7]	M	0x0 0205	y[256]	E	0	0x0 0207	z[256]	M
…	..	…	…	…	…	…	…	..	…	…	…	…	…	…
31	0x0 0204	y[255]	E	0	0x0 0206	z[255]	M	0x0 0205	y[511]	E	0	0x0 0207	z[511]	M