Un computer a ciclo singolo (che utilizza l'architettura vista sul libro) ha un clock di 8 MHz, è dotato di linee di indirizzi a 16 bit, di una memoria ram SDRAM da 16 KB per le istruzioni e una memoria SDRAM da 32 KB per i dati. Le memorie sono indirizzate a word di 16 bit. Collegata alla sola memoria dati vi è una cache a mappatura diretta con 4 line di 8 byte. Il tempo di accesso alle memorie RAM e cache sono, rispettivamente, di 60 ns e di 12 ns; l'architettura del computer permette di fare il fetch dell'istruzione e un accesso alla memoria all'interno di uno stesso ciclo di clock.
Nel computer viene eseguito un programma costituito dalle istruzioni memorizzate nella memoria delle istruzioni a partire dall'indirizzo 0x0 (vedi tabella B).
Al momento dell'esecuzione del codice gli 8 registri contengono i valori riportati in figura (vedi tabella C), la memoria dati contiene i valori riportati in figura (vedi tabella A) e la memoria cache è vuota.

Si risponda alle seguenti domande:

1. Si indichi, a parole, quale è lo scopo del programma
2. Si indichi quale è il valore che si trova nel registro R7 al termine dell'esecuzione del programma
3. Si indichi il numero di istruzioni complessive eseguite dal programma
4. Si calcoli il numero di cache hit e il numero di cache miss durante l'esecuzione del programma
5. Si calcoli quanti ms sono impiegati in totale per fare gli accessi ai dati (si ignorino i periodi di refresh)

A)

RAM Dati a 16 bit	
0x0000	610
0x0001	33
0x0002	77
0x0003	90
0x0004	11
0x0005	21
0x0006	12
0x0007	34
0x0008	6
0x0009	34
0x000A	8
0x000B	32
0x000C	75
0x000D	343
0x000E	23
0x000F	240

B)

RAM Istruzioni a 16 bit				
0x0000	LD	R0	R2	
0x0001	INC	R2	R2	
0x0002	DEC	R3	R3	
0x0003	BRZ	R3		6
0x0004	LD	R1	R2	
0x0005	SUB	R4	R1	R0
0x0006	BRN	R4		2
0x0007	MOV	R0	R1	
0x0008	JMP	R6		
0x0009	MOV	R7	R0	

C)

Valori nei registri	
R0	42
R1	9
R2	7
R3	6
R4	13
R5	17
R6	1
R7	42

TABLE 8-8
Instruction Specifications for the Simple Computer

Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move A	0000000	MOVA	RD, RA	R[DR] ← R[SA]*	N, Z
Increment	0000001	INC	RD, RA	R[DR] ← R[SA] + 1*	N, Z
Add	0000010	ADD	RD, RA, RB	R[DR] ← R[SA] + R[SB]*	N, Z
Subtract	0000101	SUB	RD, RA, RB	R[DR] ← R[SA] − R[SB]*	N, Z
Decrement	0000110	DEC	RD, RA	R[DR] ← R[SA] − 1*	N, Z
AND	0001000	AND	RD, RA, RB	R[DR] ← R[SA] ∧ R[SB]*	N, Z
OR	0001001	OR	RD, RA, RB	R[DR] ← R[SA] ∨ R[SB]*	N, Z
Exclusive OR	0001010	XOR	RD, RA, RB	R[DR] ← R[SA] ⊕ R[SB]*	N, Z
NOT	0001011	NOT	RD, RA	R[DR] ← $\overline{\text{R[SA]}}$*	N, Z
Move B	0001100	MOVB	RD, RB	R[DR] ← R[SB]*	
Shift Right	0001101	SHR	RD, RB	R[DR] ← sr R[SB]*	
Shift Left	0001110	SHL	RD, RB	R[DR] ← sl R[SB]*	
Load Immediate	1001100	LDI	RD, OP	R[DR] ← zf OP*	
Add Immediate	1000010	ADI	RD, RA, OP	R[DR] ← R[SA] + zf OP*	N, Z
Load	0010000	LD	RD, RA	R[DR] ← M[SA]*	
Store	0100000	ST	RA, RB	M[SA] ← R[SB]*	
Branch on Zero	1100000	BRZ	RA, AD	if (R[SA] = 0) PC ← PC + se AD, if (R[SA] ≠ 0) PC ← PC + 1	N, Z
Branch on Negative	1100001	BRN	RA, AD	if (R[SA] < 0) PC ← PC + se AD, if (R[SA] ≥ 0) PC ← PC + 1	N, Z
Jump	1110000	JMP	RA	PC ← R[SA]*	

* For all of these instructions, PC ← PC + 1 is also executed to prepare for the next cycle.

0		LD	R0	R2	
1	LOOP	INC	R2	R2	
2		DEC	R3	R3	
3		BRZ	R3	6	:STOP
4		LD	R1	R2	
5		SUB	R4	R1	R0
6		BRN	R4	2	
7		MOV	R0	R1	
8		JMP	R6		:LOOP
9	STOP	MOV	R7	R0	

CACHE a 4 righe con linee 8 byte

RAM	32 KB	sono 16 K locazioni a 16bit	14 bit
Cache	linee da 8 byte, ovvero 4 locazioni		4K tag diversi

RAM a 16 bit		CACHE INDEX
0x0	610	0
0x1	33	0
0x2	77	0
0x3	90	0
0x4	11	1
0x5	21	1
0x6	12	1
0x7	34	1
0x8	6	2
0x9	34	2
0xA	8	2
0xB	32	2
0xC	75	3
0xD	343	3
0xE	23	3
0xF	240	3

R0	R1	R2	R3	R4	R5	R6	R7
42	9	7	6	13	17	1	42

OPS	RUN				JMP	R0	R1	R2	R3	R4	R5	R6	R7	CACHE								
1	LD	R0	R2			M[7] = 34								MISS		INSERT	1		11	21	12	34
1	INC	R2	R2					8														
1	DEC	R3	R3						5													
1	BRZ	R3		:STOP	NO																	
1	LD	R1	R2				M[8] = 6							MISS		INSERT	2		6	34	8	32
1	SUB	R4	R1	R0						R1-R0=-28												
1	BRN	R4	2		SI																	
0	MOV	R0	R1		X																	
1	JMP	R6		:LOOP	SI																	
1	INC	R2	R2					9														
1	DEC	R3	R3						4													
1	BRZ	R3		:STOP	NO																	
1	LD	R1	R2				M[9] = 34							HIT						X		
1	SUB	R4	R1	R0						R1-R0=0												
1	BRN	R4	2		NO																	
1	MOV	R0	R1																			
1	JMP	R6		:LOOP	SI																	
1	INC	R2	R2					10														
1	DEC	R3	R3						3													
1	BRZ	R3		:STOP	NO																	
1	LD	R1	R2				M[10]=8							HIT							X	
1	SUB	R4	R1	R0						R1-R0=-26												
1	BRN	R4	2		SI																	
0	MOV	R0	R1		X																	
1	JMP	R6		:LOOP	SI																	
1	INC	R2	R2					11														
1	DEC	R3	R3						2													
1	BRZ	R3		:STOP	NO																	
1	LD	R1	R2				M[11]=32							HIT								X
1	SUB	R4	R1	R0						R1-R0=-2												
1	BRN	R4	2		SI																	
0	MOV	R0	R1		X																	
1	JMP	R6		:LOOP	SI																	
1	INC	R2	R2					12														
1	DEC	R3	R3						1													
1	BRZ	R3		:STOP	NO																	
1	LD	R1	R2				M[12]=75							MISS		INSERT	3		75	343	23	240
1	SUB	R4	R1	R0						R1-R0=41												
1	BRN	R4	2		NO																	
1	MOV	R0	R1			75																
1	JMP	R6		:LOOP	SI																	
1	INC	R2	R2					13														
1	DEC	R3	R3						0													
1	BRZ	R3		:STOP	SI																	
1	MOV	R7	R0										75									

ISTRUZIONI ESEGUITE
42

il codice implementa una funzione che trova il valore massimo all'interno di un array di valori

MISS	HIT	TEMPO	
3	3	3 * 60 + 3 * 12 =	216 ns