Svolgimento

Vogliamo fare la somma dei numeri di un vettore.

- Il vettore ha n elementi
- Il primo elemento si trova alla riga 27 della memoria
- Nella riga 27-1 è scritta la lunghezza del vettore (che ha 4 elementi)
- Il risultato della somma va scritto nella riga 32
- Nel registro R0 è scritto il numero 26
- Nel registro R1 è scritto il numero 32

Registro	Valore
0	26
1	32
2	…
3	…

Posizione in memoria	Valore
26	4
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.

Soluzione

```
// implementazione assembly senza ciclo for
// CODICE CREATO IN CLASSE

     // faccio la prima somma:
     // leggo primo e secondo elemento
     // e li sommo
     // uso R0 come indirizzo da usare per gli accessi in memoria
INC   R0    R0          // r0 = r0+1     r0 = 27
LD    R2    R0          //               r2 = 11
INC   R0    R0          // r0 = r0+1     r0 = 28
LD    R3    R0          //               r3 = 22
ADD   R2    R2    R3    // r2 = r2+r3    r2 = 33

     // leggo terzo elemento e lo sommo
INC   R0    R0          // r0++          r0 = 29
LD    R4    R0          //               r4 = 33
ADD   R2    R2    R4    //               r2 = 66

     // leggo quarto elemento e lo sommo
INC   R0    R0          // r0++          r0 = 30
LD    R3    R0          //               r3 = 44
ADD   R5    R2    R3

     // salvo il risultato in memoria
ST    R1    R5
```

Registro	Valore
0	26
1	32
2	
3	
4	
5	
6	
7	

Posizione in memoria	Valore
26	4
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.

Soluzione

```
// implementazione assembly senza ciclo for
// SOLUZIONE UGUALE ALLA PRECEDENTE MA FATTA DAL PROF E COMMENTATA

INC R4, R0            // ho creato il numero 27
                      // in R4 abbiamo 27

LD R5, R4             // carico in R5 il valore alla
                      // riga 27 della memoria
                      // in R5 abbiamo 11

INC R4, R4            // incrementiamo l'indice
                      // in R4 abbiamo 28

LD R6, R4             // carico in R6 il valore della
                      // riga 28 della memoria
                      // in R6 abbiamo 22

ADD R5, R5, R6        // sommiamo in R5 il valore 22
                      // in R5 abbiamo 11+22 = 33

INC R4, R4            // in R4 abbiamo 29
LD R6, R4             // in R6 abbiamo 33
ADD R5, R5, R6        // in R5 abbiamo 33 + 33 = 66

INC R4, R4            // in R4 abbiamo 30
LD R6, R4             // in R6 abbiamo 44
ADD R5, R5, R6        // in R5 abbiamo 66 + 44 = 110

ST R1, R5             // salviamo il risultato
                      // nella riga 32 della memoria
```

Registro	Valore
0	26
1	32
2	
3	
4	
5	
6	
7	

Posizione in memoria	Valore
26	4
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.

Esercizio

Vogliamo fare la somma dei numeri di un vettore.

- Il vettore ha n elementi
- Il primo elemento si trova alla riga 27 della memoria
- Nella riga 27-1 è scritta la lunghezza del vettore (che ha 4 elementi)
- Il risultato della somma va scritto nella riga 32
- Nel registro R0 è scritto il numero 26
- Nel registro R1 è scritto il numero 32

VOGLIAMO USARE UN CICLO

Registro	Valore
0	26
1	32
2	…
3	…

Posizione in memoria	Valore
26	200
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.

Soluzione

```
// SOLUZIONE FATTA IN CLASSE
// pseudo codice java e traduzione dei comandi
int num = 200;                    LD
int[] pippo = int[ num ];
int somma = 0;                    LDI
for( int i=0; i< num; i++ ) {     LDI   i=0
                                  SUB   i - num
                                  BRN   se i >=num salta a "qui"
                                  INC   i++
     int elem = pippo[ i ];       LD
     somma = somma + elem;        ADD
}                                 JMP   riga 5
qui

--------------------
// implementazione assembly con ciclo

LD R2, R0        // num = M[r0] = 200
LDI R3, 0        // somma = 0

LDI R4, 0        // i=0
SUB R5, R2, R4   // r5 = num-i
BRZ R5, 7        // se i==num salta il ciclo
INC R0, R0       // in r0 abbiamo 27
LD R6, R0        // elem = pippo[r0]
ADD R3, R3, R6   // somma += elem (r6)
INC R4, R4       // i++
LDI R7, 3
JMP R7
ST R1, R3        // salvo il risultato in memoria
```

Registro	Valore
0	26
1	32
2	
3	
4	
5	
6	
7	

Posizione in memoria	Valore
26	4
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.

Soluzione

```
// SOLUZIONE FATTA DAL PROF E COMMENTATA
// pseudo codice java
int[] pippo = new int[4];

int i=0;                          // indice
int contatore = 4;                // contatore
int somma = pippo[i];             // accumulatore
contatore--;
while( contatore > 0 ) {
      i++;
      int val = pippo[i];
      somma = somma + val;
      contatore--;
}

--------------------
// implementazione assembly con ciclo

INC R4, R0            // in R4 ho indice: lo inizializzo a 27
LD R5, R4             // salvo nell'accumulatore il valore 11
LD R7, R0             // R7 contiene il contatore, parte da 4
DEC R7, R7            // R7 contiene 3

ciclo:
INC R4, R4            // in R4 abbiamo l'indice i: lo incremento
LD R6, R4             // in R6 carichiamo il valore
ADD R5, R5, R6        // in R5 aggiorniamo l'accumulatore
DEC R7, R7            // R7 contiene 2, 1, 0

BRZ R7, 2             // se R7 e' zero, esco dal ciclo (non eseguo jump)
JUMP ciclo            // re-inizio ciclo

ST R1, R5             // salvo il risultato in memoria
```

Registro	Valore
0	26
1	32
2	
3	
4	
5	
6	
7	

Posizione in memoria	Valore
26	4
27	11
28	22
29	33
30	44
31	…
32	…

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] ← $\overline{R[SA]}$*	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.