Sintesi di Reti Sequenziali

Esercizio 1. Si realizzi una rete sequenziale sincrona R con un ingresso **X** ed un'uscita **Z**. Ogni tre colpi di clock la rete riceve in ingresso sulla linea **X** i tre bit **b(t-2)**, **b(t-1)** e **b(t)**. Al ricevimento del terzo bit **b(t)** la rete deve restituire in uscita zero se il numero binario naturale formato dai tre bit **b(t-2) b(t-1) b(t)** è un numero primo, uno altrimenti. Successivamente la rete riprende il suo funzionamento dal principio. Segue un esempio di funzionamento.

t:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
X:	0	1	1	1	0	0	0	0	0
Z:	0	0	0	0	0	1	0	0	1

Esercizio 2. Si realizzi una rete sequenziale sincrona R con un ingresso **X** ed un'uscita **Z**. Ad ogni colpo di clock, R riceve un bit sulla linea **X**. I primi due bit $\mathbf{b_0}$ e $\mathbf{b_1}$ ricevuti sulla linea **X** indicano alla rete R quante volte riconoscere la sottosequenza **11**. Al termine del riconoscimento, la rete restituisce **1** sulla linea **Z** e riconosce una nuova sequenza. Nel caso in cui sia $\mathbf{b_0}$ che $\mathbf{b_1}$ sono uguali a zero, la rete non dovrà riconoscere nessuna sequenza e, quindi, leggerà una nuova coppia $\mathbf{b_0}$ $\mathbf{b_1}$. Segue un esempio di funzionamento.

t:	0	1	2	3	4	5	6	7	8
x(t):	1	0	0	1	1	1	0	1	1
z(t):	0	0	0	0	0	0	0	0	1

Esercizio 3. Si realizzi una rete sequenziale sincrona R con un ingresso **X** ed una uscita **Z**. La rete deve riconoscere sequenze di bit **S=αβγ**, in cui **α** e **γ** sono sottosequenze di 2 bit e **β** è un bit. La rete si aspetta di riconoscere la sequenza **γ= (α+β)%4**. La rete restituisce 1 dopo aver riconosciuto una sequenza S corretta, 0 altrimenti. Si noti che, in ogni caso, la sottosequenza **γ** va letta fino alla fine (leggendo entrambi i suoi bit) prima di produrre 1 o 0 in uscita e prepararsi a leggere una nuova sequenza. Segue un esempio di possibile funzionamento di R:

t:	0	1	2	3	**4**	5	6	7	**8**	**9**
x:	1	1	1	0	0	1	0	1	0	1
z:	0	0	0	0	**1**	0	0	0	0	**0**

La prima sottosequenza **α** è x(t0) x(t1) = 11 e il bit **β**= x(t2) = 1. Nei due istanti seguenti la rete riconosce la sequenza **γ** = x(t3) x(t4) = 00 corretta, restituisce 1 e si prepara a leggere una nuova sequenza. La seconda sottosequenza **α** è x(t5) x(t6) =10 e il bit **β**= x(t7) = 1. Nei due istanti seguenti la rete legge la sequenza di due bit x(t8) x(t9) = 01 non corretta, restituisce 0 e si prepara a leggere una nuova sequenza.

Esercizio 4. Si realizzi una rete sequenziale sincrona R con due linee di ingresso **A** e **B** ed una linea di uscita **Z**. Ad ogni colpo di clock, R riceve un bit sulla linea **A** e un bit sulla linea **B**. Il calcolo si ferma quando R avrà ricevuto su **B** esattamente quattro bit ad 1 e dovrà restituire in uscita 1 solo se la stringa (di quattro bit) che si forma su **A** in corrispondenza dei bit a 1 di **B** è palindroma. Segue un esempio di funzionamento di R.

t:	0	1	2	3	4	5	6	7
A(t):	1	0	0	0	1	0	1	1
B(t):	1	0	1	0	0	1	0	1
z(t):	0	0	0	0	0	0	0	1

Esercizio 5. Si realizzi una rete sequenziale sincrona R con un ingresso **X** ed una uscita **Z**. Ogni quattro colpi di clock la rete riceve in ingresso sulla linea **X** i quattro bit **b(t-3)**, **b(t-2)**, **b(t-1)** e **b(t)**. Al ricevimento del quarto bit **b(t)** la rete deve restituire in uscita 1 se il numero binario formato dai quattro bit **b(t-3) b(t-2) b(t-1) b(t)** è il successore di un numero primo e 0 altrimenti (si assuma che **b(t)** sia il bit meno significativo di tale numero). Successivamente la rete riprende il suo funzionamento dal principio. Segue un possibile funzionamento di R:

t:	0	1	2	3	4	5	6	7	8	9	10	11
X:	0	0	0	1	0	1	0	0	1	0	0	1
Z:	0	0	0	0	0	0	0	1	0	0	0	0

Microprogrammazione

Esercizio 1. Estendere il set di istruzioni della macchina a registri con l'operazione **MODRAM Ri, X**. L'operazione modifica le prime n locazioni della RAM, se **n** è il valore contenuto in **Ri**, a partire dalla locazione di memoria **X** (vale a dire le locazioni di indirizzo **Y=X, X+1,...,X+n–1**) come segue: se **M[Y]** memorizza un numero pari, allora **M[Y]** deve essere sostituito da **M[Y]–1**, altrimenti se **M[Y]** memorizza un numero dispari, allora **M[Y]** deve essere sostituito da **M[Y]+1**. ***Esempio:*** Sia X=0, n=2 e M[0]=11 e M[1]=8. Al termine dell'esecuzione dell'istruzione MODRAM X le locazioni M[0] ed M[1] della RAM dovranno contenere i seguenti valori: M[0]=12, M[1]=7.

Esercizio 2. Estendere il set di istruzioni della macchina a registri con l'operazione **SUMM Ri, Rj, Rk, X**. L'operazione restituisce in **Ri** la somma degli elementi maggiori di **k** presenti nel vettore di dimensione **n** memorizzato a partire dall'indirizzo **X**, dove **k** è il valore contenuto in **Rj** ed **n** è il valore contenuto in **Rk**.

Esercizio 3. Estendere il set di istruzioni della macchina a registri con l'operazione **CMP Ri**, che restituisce in **Ri** il numero degli elementi presenti in RAM ciascuno dei quali è maggiore del suo predecessore.

Esercizio 4. Estendere il set di istruzioni della macchina a registri con l'operazione **CNT Ri, Rj, X**. che, dati due vettori **V** e **W**, entrambi di dimensione pari al valore contenuto in **Rj** e memorizzati in RAM a partire dall'indirizzo **X** ed **X+1**, rispettivamente, restituisce in **Ri** il numero di coppie **(V[i], W[i])** costituite da elementi che occupano la stessa posizione nei rispettivi vettori e tali che **V[i] > W[i]**.

Esercizio 5. Estendere il set di istruzioni della macchina a registri con l'operazione **SUMHALF Ri, X**. A partire dalla locazione di memoria **M[X+1]** è memorizzato un vettore **V** di lunghezza pari ad **L**, che è il valore contenuto in **Ri**. L'istruzione restituisce nella locazione **M[X+L+1]** la somma degli elementi contenuti nella prima metà del vettore (da **M[X+1]** a **M[X+L/2]**) e nella locazione **M[X+L+2]** la somma degli elementi contenuti nella seconda metà del vettore (da **M[X+L/2+1]** a **M[X+L]**).

Esempio: Supponiamo che X=500 e L=10, e sia V=[3,2,4,1,1,5,2,2,1,3] il vettore di lunghezza 10 memorizzato a partire dall'indirizzo X+1=501. Al termine dell'esecuzione di SUMHALF X, la locazione M[511] conterrà il valore 11 (somma degli elementi nella prima metà di V: 3+2+4+1+1=11), mentre la locazione M[512] conterrà il valore 13 (somma degli elementi nella seconda metà di V: 5+2+2+1+3=13).

Assembly

Esercizio 1. Scrivere un programma assembly che, dati due vettori V1 e V2, stampi “vero” se V1 è uguale a V2 capovolto e stampi “falso” altrimenti.

Esercizio 2. Data una matrice quadrata **M** di interi a 16 bit e un numero **k**, scrivere un programma assembly che stampi “vero” se la somma degli elementi della matrice triangolare inferiore della matrice ad esclusione della diagonale è superiore a **k** e stampi “falso” altrimenti.

Segue un esempio:

M:

1	3	14	-3	3
2	5	0	9	-6
-9	-7	4	8	-2
0	2	5	-9	-1
11	3	-4	0	9

k = 5

Il programma stamperà “falso” perché la somma degli elementi evidenziati è pari a 3 < 5.

Esercizio 3. Sia dato un vettore contenente informazioni riguardanti le temperature rilevate in alcune città. In particolare, per ogni città, nel vettore sono memorizzati l'iniziale della città (1 byte) e due interi a 16 bit indicanti, rispettivamente, la temperatura minima e quella massima. Scrivere un programma assembly che, dato tale vettore, stampi l'iniziale della città con il più alto valore di escursione termica (differenza tra temperatura massima e minima) e, a parità di escursione, la prima città in ordine alfabetico.

Esempio. Si consideri il vettore V = [['m', 2, 7],['a', 2, 8],['r', 1, 9],['p', -2, 6]] [1] contenente informazioni su 4 città, Milano, Ancona, Roma e Perugia. Il programma stamperà il valore 'p' dato che Perugia ha il massimo valore di escursione termica (8°), come Roma, ma precede quest'ultima in ordine alfabetico.

[1] V: db 'm', 2h, 0h, 7h, 0h, 'a', 2h, 0h, 8h, 0h, 'r', 1h, 0h, 9h, 0h, 'p', 0FEh, 0FFh, 6h, 0h

Esercizio 4. Sia dato un intero **k** a 32 bit e un vettore **V** contenente n indirizzi di memoria, a partire da ciascuno dei quali è memorizzata una matrice 4x4 di interi a 16 bit. Una matrice è considerata *valida* se la somma degli elementi sulla diagonale secondaria è inferiore a **k**. Scrivere un programma assembly che stampi “vero” se tutte le matrici il cui indirizzo è in **V** sono valide, stampi “falso” altrimenti. Segue un esempio.

```
section .data
m1:     dw      5,8,25,-7,2,-11,-6,10,14,15,27,-4,-18,-31,2,-5
m2:     dw      6,3,15,-9,5,-31,30,23,12,10,34,-2,-32,-11,3,-2
m3:     dw      8,3,34,12,4,-24,-5,14,42,21,56,29,-25,-32,5,14
v:      dd      m1, m2, m3
k       equ     5
```

Il programma stamperà “vero”, infatti:

- m1 è valida (-7-6+15-8 = -6 < k),
- m2 è valida (-9+30+10-32 = -1 < k)
- m3 è valida (12-5+21-25 = 3 <k).

Esercizio 5. Scrivere una programma assembly che, dato un intero **X** (a 16 bit) e una matrice quadrata di interi a 16 bit, verifichi che la somma in valore assoluto degli elementi negativi posti in

posizione **[i,j]**, tale che o **i** e **j** sono entrambi indici dispari o **i** e **j** sono entrambi indici pari, sia inferiore ad **X**. Memorizzare l'esito della verifica in una variabile (memorizzare “1” per “vero” e “0” per “falso”).

Esempio:

M:

1	3	25	-7
2	-11	-6	10
14	15	27	-4
-8	-31	2	-5

Nell'esempio, gli elementi della matrice M che rispettano la condizione sulla posizione sono evidenziati in grigio.
Assumendo X = 50, il programma memorizzerà “1” in quanto $|(-11)+(-31)+(-5)|=47 < 50$.