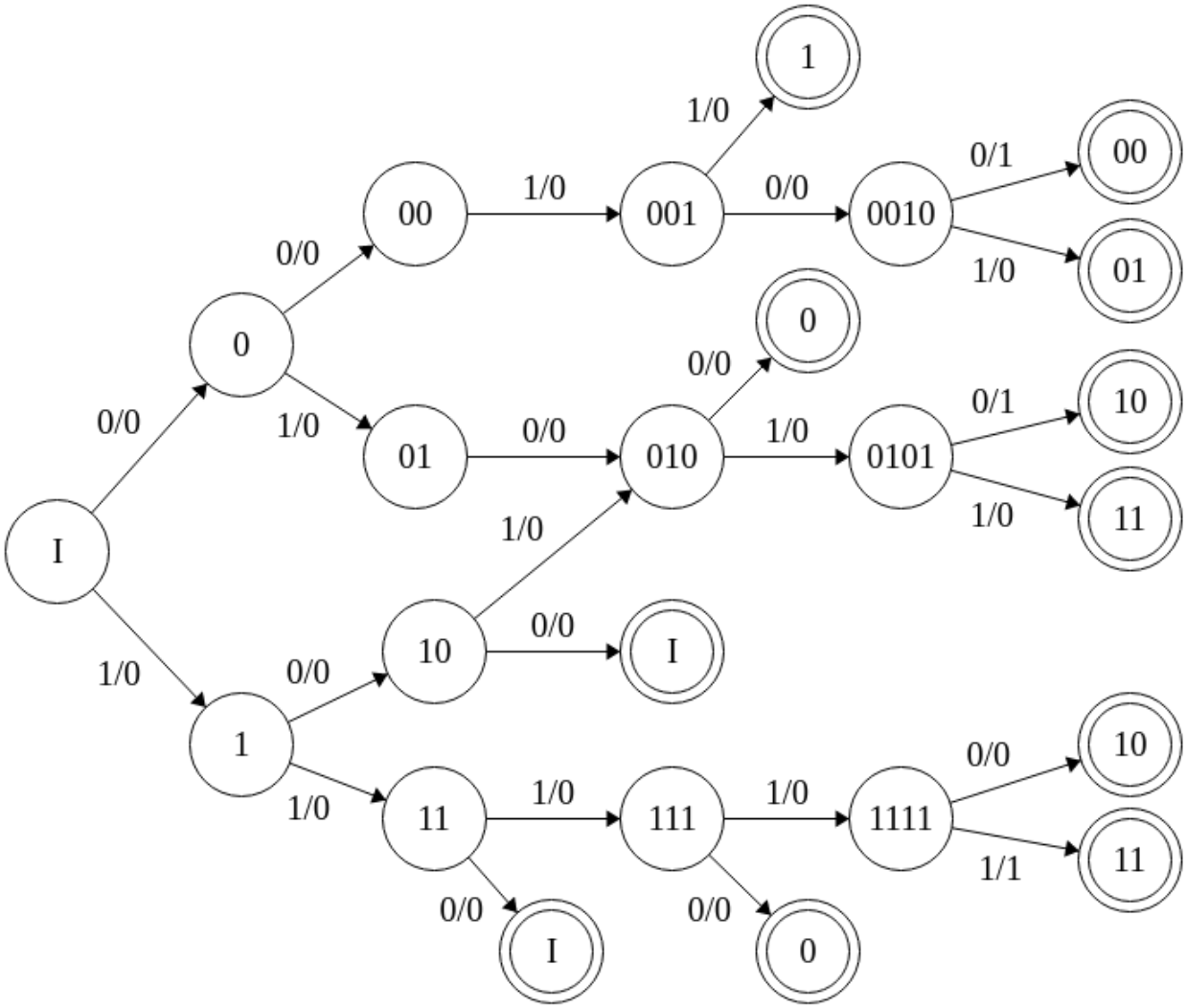
RETI LOGICHE E CALCOLATORI – 18/07/2016 – TRACCIA A – ESERCIZIO 1

Si realizzi una rete sequenziale sincrona R con un ingresso X ed una uscita Z. La rete riconosce sequenze del tipo $\mathbf{b_1b_0c_0c_1c_2}$, dove la coppia $B=b_1b_0$ e la tripla $C=c_2c_1c_0$ rappresentano due numeri in notazione binaria naturale. Si noti che l'ordine di ricezione dei bit che compongono i numeri B e C non è il medesimo. Infatti, per quanto riguarda B viene ricevuto prima il bit più significativo e poi il bit meno significativo mentre, per quanto riguarda C, il primo bit a essere ricevuto è il meno significativo. Dopo avere ricevuto i 5 bit, la rete restituisce 1 se C = B + 1 e restituisce 0 altrimenti. Successivamente la rete si appresta a riconoscere una nuova sequenza, sovrapposta alla precedente di due bit, quindi i bit c_1 e c_2 di una sequenza coincidono con i bit b_1 e b_0 della successiva. Si guardi l'esempio per maggiore chiarezza.

t:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
X:	0	1	0	0	0	1	0	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	1	…
Z:	0	0	0	0	**0**	0	0	**1**	0	0	**0**	0	0	**1**	0	0	**1**	0	0	**0**	0	0	**1**	…

Nell'esempio, la rete riceve la prima sequenza a partire dall'istante t=0 e, in particolare, riceve $b_1b_0 = 01$ e $c_2c_1c_0 = 000$, quindi B è il numero naturale 1 mentre C è il numero naturale 0. Non essendo C = B+1, la rete restituisce 0. La successiva sequenza (sovrapposta di due bit alla precedente) è ricevuta a partire dall'istante t=3. In particolare, la rete riceve $b_1b_0 = 00$ e $c_2c_1c_0 = 001$. Per tale sequenza B = 0 e C = 1, pertanto la rete restituisce 1; e così via.



RETI LOGICHE E CALCOLATORI – 18/07/2016 – TRACCIA A – ESERCIZIO 2

Estendere il set di istruzioni della macchina ad accumulatore con l'operazione **SUMV@ X,** definita come segue.
A partire dalla locazione X+1 della RAM è memorizzato un vettore **V** di L elementi, dove L è contenuto in M[X].
Ogni elemento del vettore è l'indirizzo di una locazione di memoria in cui è memorizzato un intero. L'istruzione calcola la somma del valore assoluto degli interi presenti in memoria i cui indirizzi sono presenti in **V** e restituisce tale somma nell'accumulatore.
La figura sulla destra mostra un esempio dello stato della memoria. Al termine dell'esecuzione, l'accumulatore conterrà il valore 17 ottenuto dalla somma 2+3+4+8.

X			
		:	:
1052	**L**	1052	4
	V[0]	1053	1600
	V[1]	1054	2100
	V[2]	1055	1350
	V[3]	1056	2310
		:	:
		1350	-4
		:	:
		1600	2
		:	:
		2100	-3
		:	:
		2310	8
		:	:

CODICE RTL

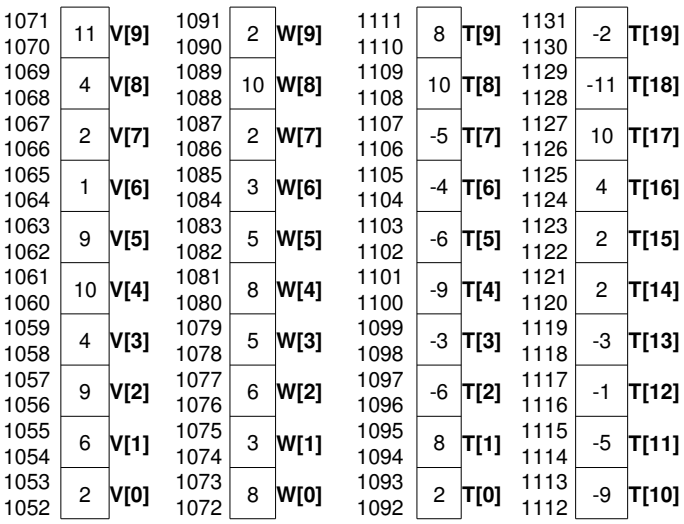
μ_1 $IR_X \rightarrow MAR$, $0 \rightarrow A$;
μ_2 $M[MAR] \rightarrow MBR$, $INCR(MAR) \rightarrow MAR$;
μ_3 $MAR \rightarrow IND$, $MBR \rightarrow T1$;
1: if $OR(T1) = 1$ **then**
μ_4 $IND \rightarrow MAR$, $INCR(IND) \rightarrow IND$;
μ_5 $M[MAR] \rightarrow MBR$;
μ_6 $MBR \rightarrow MAR$;
μ_7 $M[MAR] \rightarrow MBR$;
μ_8 $MBR \rightarrow B$;
if $MBR_{31} = 1$ **then**
μ_9 $-B \rightarrow B$;
μ_{10} $A+B \rightarrow A$, $DECR(T1) \rightarrow T1$, **goto** 1;
else
μ_{10} $A+B \rightarrow A$, $DECR(T1) \rightarrow T1$, **goto** 1;
fi
else
μ_0 $A \rightarrow AC$;
fi

RETI LOGICHE E CALCOLATORI – 18/07/2016 – TRACCIA A – ESERCIZIO 3

Scrivere una procedura assembly che riceve due vettori **V** e **W** composti entrambi da n elementi, con n pari, e restituisce un array **T** di dimensione pari a 2n, composto come segue. Per ogni *i*, se gli elementi **V**[i] e **W**[i] sono entrambi pari, questi dovranno essere inseriti in **T**, altrimenti in **T** dovranno essere inseriti gli elementi –**V**[i] e –**W**[i].

Scrivere inoltre il programma principale che invoca opportunamente la procedura descritta.

La figura sulla destra mostra un esempio dello stato della memoria assumendo che l'indirizzo di partenza del vettore **V** sia 1052, l'indirizzo di partenza del vettore **W** sia 1072, la lunghezza dei vettori sia uguale a 10 e l'indirizzo di partenza del vettore **T** sia 1092. Nell'esempio, gli elementi **V**[0] e **W**[0] sono entrambi pari (essendo uguali, rispettivamente, a 2 e a 8), per cui vengono inseriti in **T**. Gli elementi **V**[1] e **W**[1] non sono entrambi pari (in particolare, **W**[1] è un numero dispari) per cui in **T** vengono inseriti i valori -6 e -3; e così via.




```
%include "utils.nasm"

section .data

V       dw      2, 6, 9, 4, 10, 9, 1, 2, 4, 11
W       dw      8, 3, 6, 5, 8, 5, 3, 2, 10, 2
n       equ     ($-W)/2

section .bss

T       resw    2*n

section .text
extern proc
global _start

_start:

        push    dword n
        push    T
        push    W
        push    V
        call    proc

        mov     esi, 0
.l0:    cmp     esi, 2*n
        jge     .endl0
        mov     ax, [T+2*esi]
        printw  ax
        inc     esi
        jmp     .l0
.endl0:
        exit    0
```

```
section .data
V       equ     8
W       equ     12
T       equ     16
n       equ     20

section .text
proc:   push    ebp
        mov     ebp, esp
        pushad
        mov     eax, [ebp+V]
        mov     ebx, [ebp+W]
        mov     ecx, [ebp+T]
        mov     edi, [ebp+n]
        xor     esi, esi            ; i = 0

.ciclo: cmp     esi, edi
        jge     .fine

        mov     dx, [eax+2*esi]     ; dx = V[i]
        and     dx, 1               ; dx è pari?
        jnz     .dispari
        mov     dx, [ebx+2*esi]     ; dx = W[i]
        and     dx, 1               ; dx è pari?
        jnz     .dispari

        mov     dx, [eax+2*esi]     ; dx = V[i]
        mov     [ecx+4*esi], dx     ; T[2*i] = dx
        mov     dx, [ebx+2*esi]     ; dx = W[i]
        mov     [ecx+4*esi+2], dx   ; T[2*i+1] = dx

        inc     esi                 ; i++
        jmp     .ciclo
.dispari:

        mov     dx, [eax+2*esi]     ; dx = V[i]
        neg     dx                  ; dx = -dx
        mov     [ecx+4*esi], dx     ; T[2*i] = dx
        mov     dx, [ebx+2*esi]     ; dx = W[i]
        neg     dx                  ; dx = -dx
        mov     [ecx+4*esi+2], dx   ; T[2*i+1] = dx
        inc     esi
        jmp     .ciclo
.fine:
        popad
        mov     esp, ebp
        pop     ebp
        ret     16
```