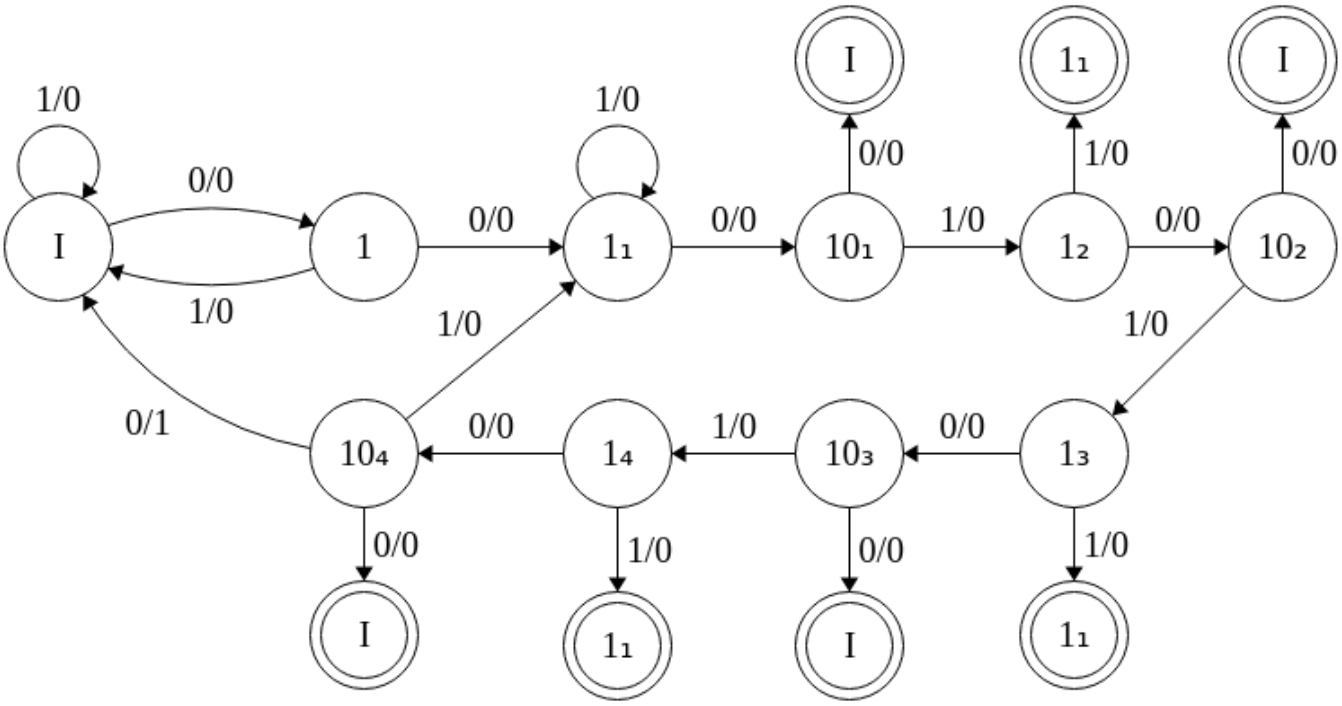
# PROVA SCRITTA DI RETI LOGICHE E CALCOLATORI DEL 20/06/2016 – TRACCIA A

**ESERCIZIO 1**:
Si realizzi una rete sequenziale sincrona R con un ingresso X ed una uscita Z. La rete riconosce sequenze del tipo **1(10)ⁿ0**, dove n è un numero intero maggiore di 0 e multiplo di 4. Quindi i valori ammissibili per n sono nell'insieme {4,8,12,16,20,...}. Non appena la rete riconosce una sequenza valida, restituisce 1 e riprende il proprio funzionamento dal principio.
Si guardi l'esempio per maggiore chiarezza.

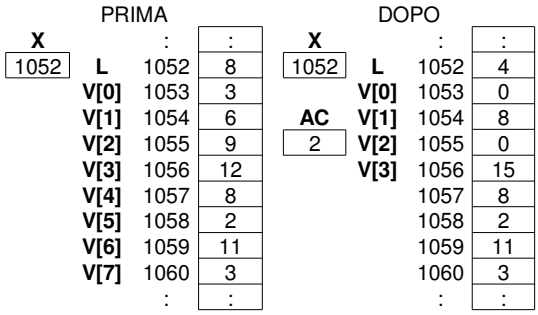
| **t:** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **X:** | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |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

Nell'esempio, la rete riceve la prima sequenza valida a partire dall'istante t=11, infatti in tale istante di tempo la rete riceve la sequenza di start “1”, negli istanti da 12 a 19 riceve quattro volte consecutive la sequenza “10” e nell'istante 20 riceve la sequenza di stop “0”. Quindi all'istante t=21 riprende il proprio funzionamento. Si noti che la sequenza “1101” ricevuta negli istanti da 1 a 4 non rappresenta una sequenza valida in quanto in questo caso n=1.



# PROVA SCRITTA DI RETI LOGICHE E CALCOLATORI DEL 20/06/2016 – TRACCIA A

**ESERCIZIO 2**: Estendere il set di istruzioni della macchina ad accumulatore con l'operazione **SUMH 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] ed è un numero pari.
L'istruzione modificherà il vettore come segue: per ogni elemento **V**[i] della prima metà del vettore (tale che i=0,…,n/2−1), **V**[i] viene posto a **V**[i]+**V**[n/2+i] se la condizione **V**[i]>**V**[n/2+i] è vera, mentre **V**[i] viene posto a 0 se la predetta condizione è falsa.
Al termine dell'istruzione la dimensione del vettore memorizzata in M[X] dovrà essere posta uguale a n/2 e l'accumulatore dovrà contenere il numero di elementi per cui la condizione è stata soddisfatta.
La figura sulla destra mostra un esempio dello stato della memoria e dei registri prima e dopo l'esecuzione dell'istruzione.




```
μ1      IRX → IND1, 0 → AC;
μ2      IND1 → MAR, INC(IND1) → IND1;     //IND1: puntatore al primo elemento
μ3      M[MAR] → MBR, IND1 → B;
μ4      MBR → T1;
μ5      SHR(T1) → T1;          //T1: la lunghezza di metà array
μ6      T1 → A;
μ7      A + B → IND2;                     //IND2: puntatore all'elemento centrale
   1: if(OR(T1) = 1) then                 // fin quando non ho raggiunto metà vettore
μ8         IND2 → MAR;
μ9         M[MAR] → MBR, IND1 → MAR;
μ10        MBR → A, M[MAR] → MBR;         // A contiene V[i+n/2]
μ11        MBR → B;                       // B contiene V[i], IND1 resta in MAR
μ12        A − B → T2;
           if(T231 = 1) then              // se V[i+n/2] < V[i], ossia A − B < 0
μ13           A + B → MBR, INC(AC) → AC;
           else
μ14           0 → MBR;
           fi
μ15        MBR → M[MAR], INC(IND1) → IND1, INC(IND2) → IND2, DEC(T1) → T1, goto 1;
       else
μ0         ϕ;
       fi
```

# PROVA SCRITTA DI RETI LOGICHE E CALCOLATORI DEL 20/06/2016 – TRACCIA A

**ESERCIZIO 3**: Scrivere una procedura assembly che riceve due vettori **V** e **W** composti entrambi da n elementi, con n pari, e modifica l'array **W** come di seguito specificato:

a) per ogni elemento **W**[i] della prima metà del vettore (tale che i=0, …,n/2−1), **W**[i] viene posto a **W**[i]+**V**[n/2+i] se la condizione **W**[i]>**V**[n/2+i] è vera, mentre **W**[i] viene posto a 0 se la predetta condizione è falsa.

b) per ogni elemento **W**[i] della seconda metà del vettore (tale che i=n/2,…,n−1), **W**[i] viene posto a **W**[i]+**V**[i−n/2] se la condizione **W**[i]>**V**[i−n/2] è vera, mentre **W**[i] viene posto a 0 se la predetta condizione è falsa.

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

PRIMA

| | V | W |
|---|---|---|
| 0 | 2 | 8 |
| 1 | 6 | 3 |
| 2 | 9 | 6 |
| 3 | 4 | 5 |
| 4 | 10 | 9 |
| 5 | 9 | 5 |
| 6 | 1 | 3 |
| 7 | 2 | 2 |
| 8 | 4 | 10 |
| 9 | 11 | 12 |

DOPO

| | V | W |
|---|---|---|
| 0 | 2 | 0 |
| 1 | 6 | 4 |
| 2 | 9 | 8 |
| 3 | 4 | 9 |
| 4 | 10 | 0 |
| 5 | 9 | 7 |
| 6 | 1 | 0 |
| 7 | 2 | 0 |
| 8 | 4 | 14 |
| 9 | 11 | 22 |

```
%include "utils.nasm"
section .data
  v dw 2, 6, 9, 4, 10, 9, 1, 2, 4, 11
  w dw 8, 3, 6, 5, 9, 5, 3, 2, 10, 12
  n equ ($-w)/2
section .text
global _start
extern somma
_start:
  push n
  push w
  push v
  call somma

  xor esi, esi
ciclo:
  cmp esi, n
  jge esci
  printw word [w+esi*2]
  inc esi
  jmp ciclo

esci:
  exit 0
```

```
%include "utils.nasm"
section .data
  V equ 8
  W equ 12
  n equ 16

section .text
global somma
somma:
  push ebp
  mov ebp, esp
  pushad
  ;
  ;prima metà
  ;
  mov eax, [ebp+V]
  mov ebx, [ebp+W]
  mov edi, [ebp+n]        ; edi = n
  add eax, edi            ; V+n/2*2
  shr edi, 1              ; edi = n/2
  xor esi, esi            ; i = 0
ciclo_1:
  cmp esi, edi            ; (i < n/2)?
  jge fine_ciclo_1
  mov cx, [ebx+esi*2]     ; cx = W[i]
  mov dx, [eax+esi*2]     ; bx = V[n/2+i]
  cmp cx, dx              ; (W[i] > V[n/2+i])?
  jle cond_falsa_1
  add cx, dx              ; x = W[i]+V[n/2+i]
  jmp avanti_1
cond_falsa_1:
  xor cx, cx              ; x = 0
avanti_1:
  mov [ebx+esi*2], cx     ; W[i] = x
  inc esi
  jmp ciclo_1
fine_ciclo_1:
  ;
  ; seconda metà
  ;
  mov eax, [ebp+V]        ; V
  lea ebx, [ebx+edi*2]    ; W+n/2*2
  xor esi, esi            ; i = 0
ciclo_2:
  cmp esi, edi            ; (i < n/2) ?
  jge fine_ciclo_2
  mov cx, [ebx+esi*2]     ; cx = V[i]
  mov dx, [eax+esi*2]     ; dx = W[i+n/2]
  cmp cx, dx              ; (W[i+n/2] > V[i])?
  jle cond_falsa_2
  add cx, dx              ; x = W[i+n/2] + V[i]
  jmp avanti_2
cond_falsa_2:
  xor cx, cx              ; x = 0
avanti_2:
  mov [ebx+2*esi], cx     ; W[i] = x
  inc esi
  jmp ciclo_2
fine_ciclo_2:
  popad
  pop ebp
  ret 12
```