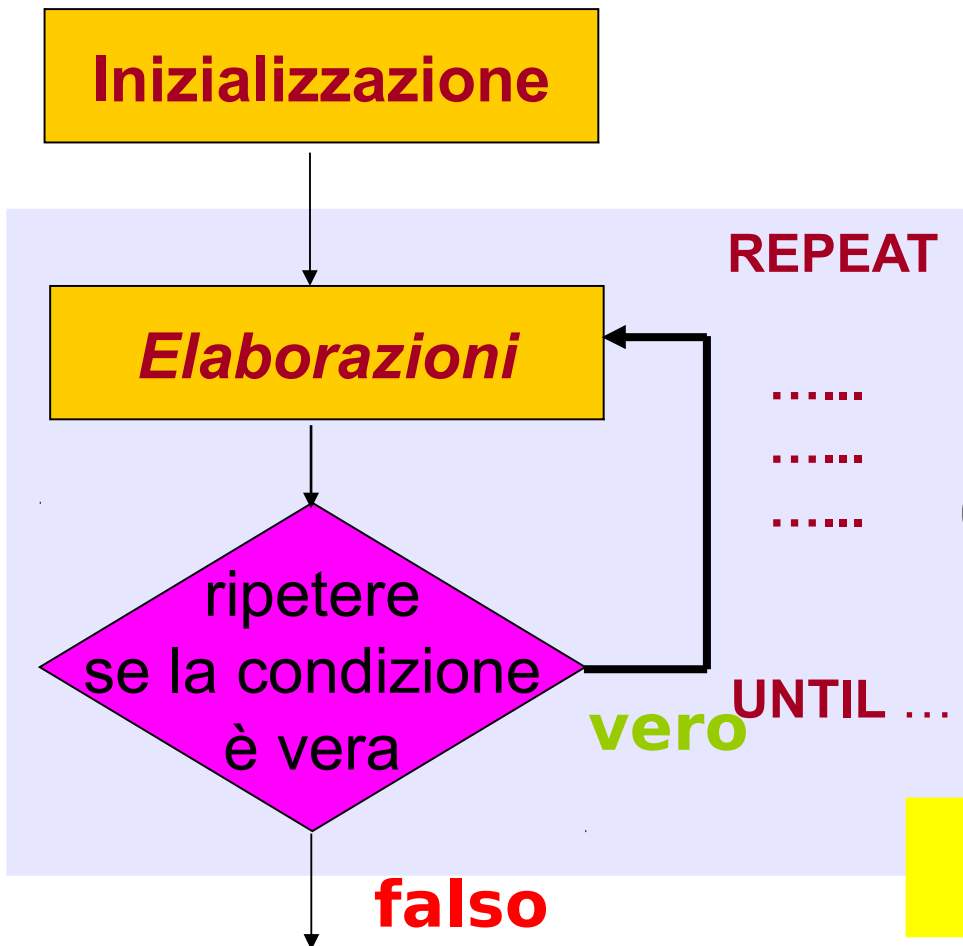


Laboratorio CPU-2 ITERAZIONI

Ivlin Stoianov, Alessia Ceccato
www.stoianov.it

CICLI (1) Ripetere un'elaborazione finché la condizione **C** è **VERA**

Esempio: Sottrarre un numero k
da un altro N **finché** $N \geq 0$



```
N: INT 15
```

```
K: INT 4
```

```
LOAD R0 N
```

```
LOAD R1 K
```

```
SUB R0 R1 % N=N-K
```

```
COMP R0 0 % Se R0 >= 0 ..
```

```
BRGE Pos % .. ripetere
```

```
STORE R0 N
```

```
STOP
```

Pos:

Se $N < 0$?

L'algorithmo non è corretto per tutti i dati possibili

Esercizio

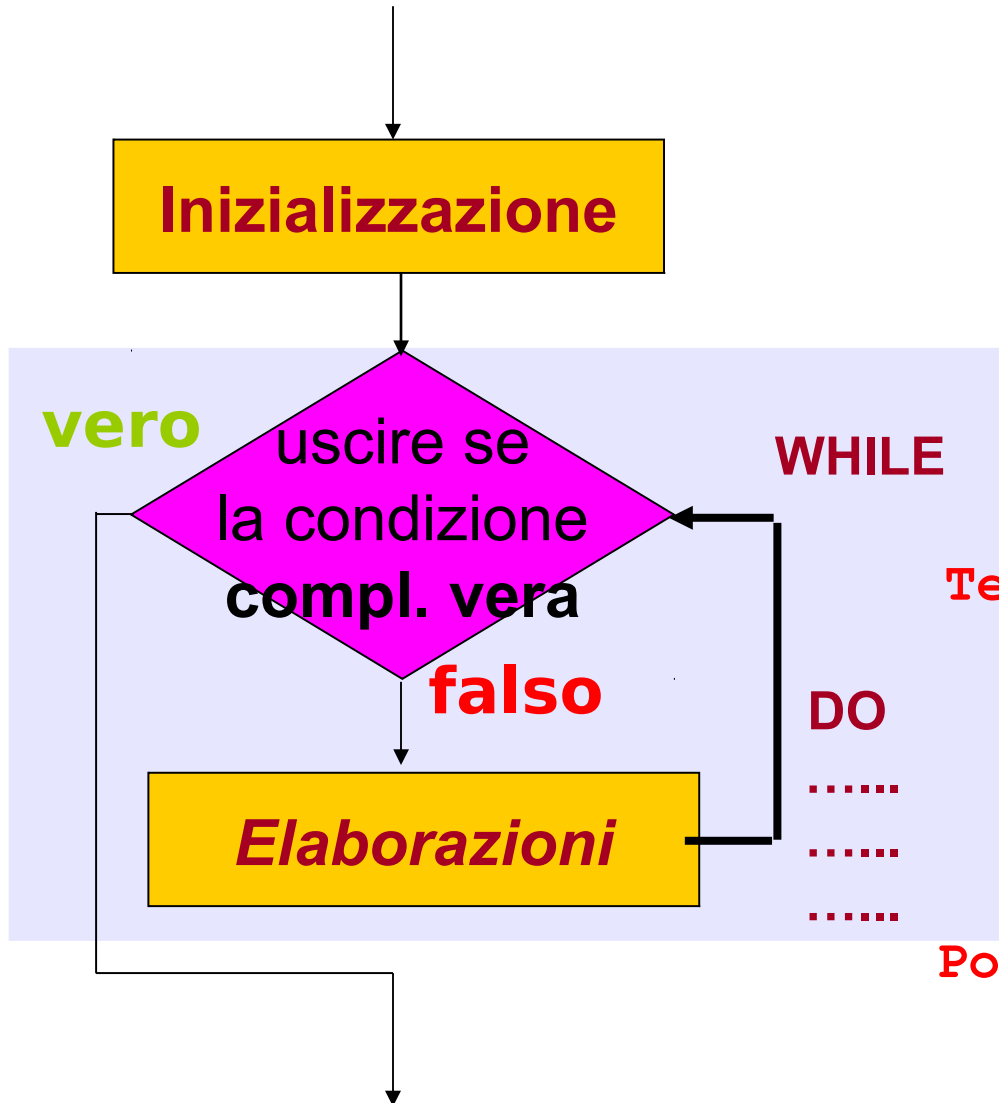
Calcolare il numero di bit necessarie di codificare con codice binario un numero N

```
N: INT 10;           Numero da codificare  
L: INT;             num-bits per il codice
```

```
LOAD R1 N;  
LOAD R0 0;  
Ripeti:DIV R1 2;     N=N/2  
ADD R0 1;           L=L+1  
COMP R1 0;  
BRGT Ripeti;  
STORE R0 L;  
STOP;
```

CICLI (2) Ripetere un'elaborazione solo se e finché la condizione **C** è VERA

Esempio: implementare la funzione "resto" della divisione del numero N ad un numero k

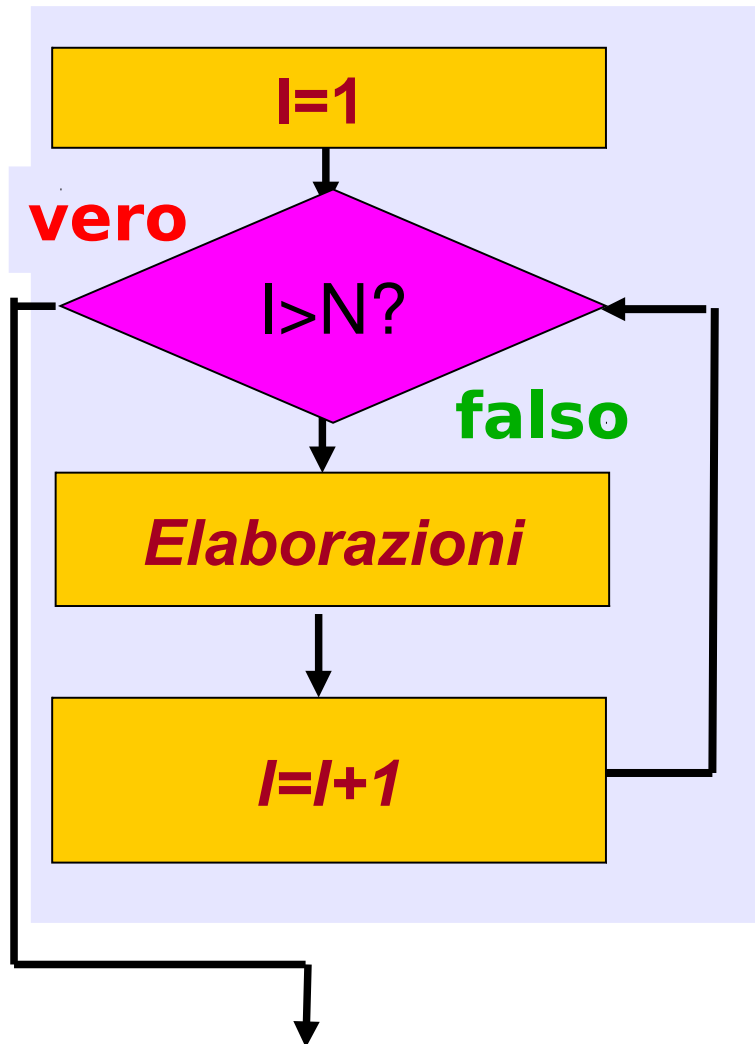


```
N: INT 15;
K: INT 6;
Resto: INT;

LOAD R0 N
LOAD R1 K
Test: COMP R0 R1 % Se R0<R1 ..
        BRLT Post % .. Uscire
SUB R0 R1 % N=N-K
        Branch Test % . ripetere
Post: STORE R0 Resto;
STOP
```

CICLI (3a) Contatore progressivo

FOR I = 1 TO N



```
N: INT 4 % Numero elabor.
```

```
LOAD R1 N
```

```
LOAD R0 1 % contatore
```

```
COMP R0 R1 % Se R0<0 ..
```

```
BRGT Usc % .. Uscire
```

```
%% ELABORAZIONI %%
```

```
ADD R0 1 % contat.+1
```

```
BRANCH Test % → testa
```

```
STOP
```

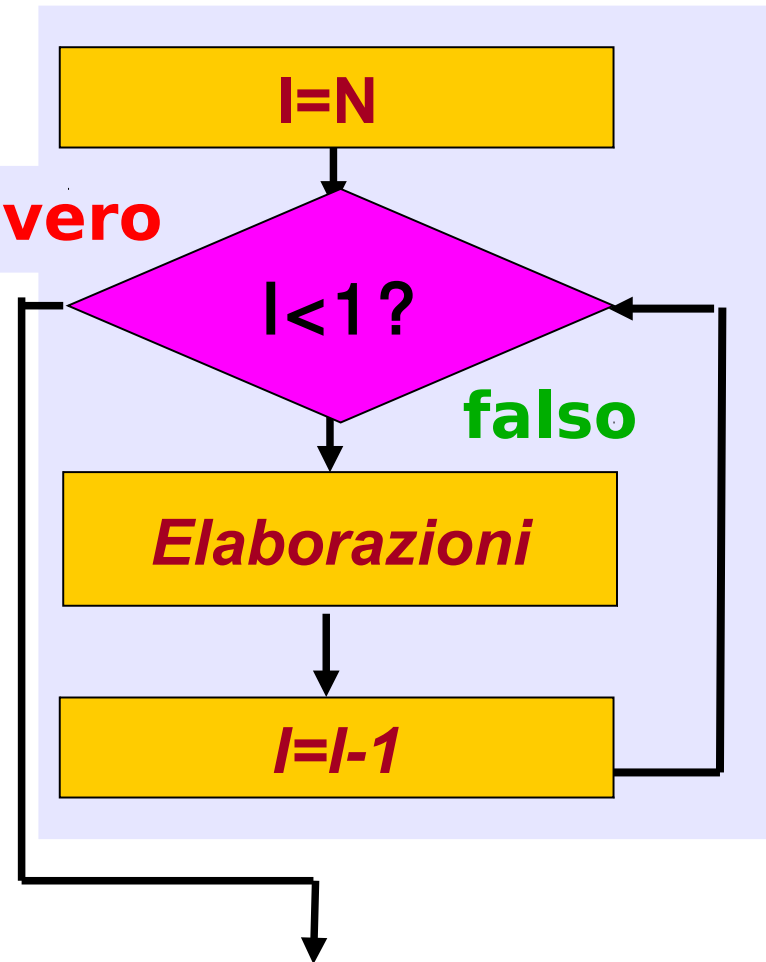
Test:

Usc:

Esercizio: fare la somma dei numeri 1, 2, .. N

CICLI (3b) Contatore regressivo

FOR I = N TO 1



```
N: INT 4 % Numero elabor.
```

```
LOAD R0 N % contatore
```

```
COMP R0 1 % Se R0<0 ..
```

```
BRLT Usc % .. Uscire
```

```
%% ELABORAZIONI %%
```

```
SUB R0 1 % contat.-1
```

```
BRANCH Test $ → testa
```

```
STOP
```

testa:

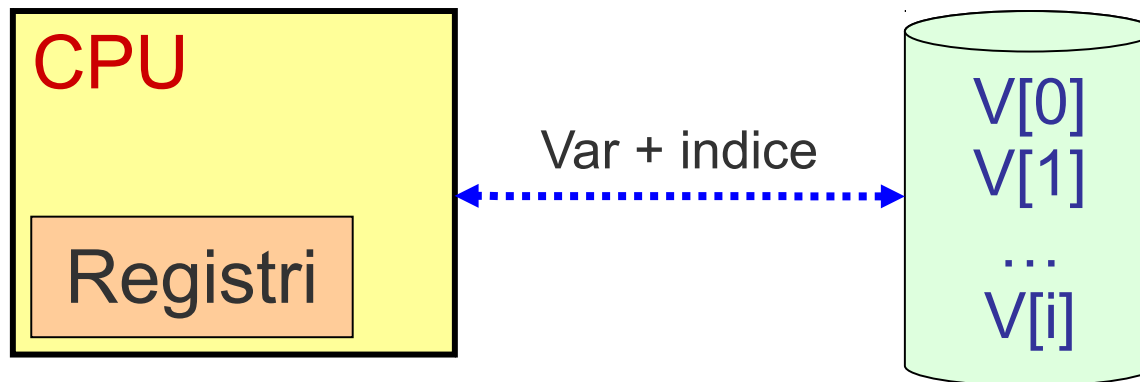
Usc:

13-for-back

Strutture dati: Vettori

- Per poter elaborare più valori, occorre memorizzarne in **strutture dati**. L'efficienza di un algoritmo dipende dalla struttura utilizzata.
- La struttura più semplice è il **vettore**, nel quale **N** valori sono localizzati in **N** successive celle di memoria (*struttura lineare*).
- Un vettore **X** (o **X[]**) è localizzato nella RAM partendo da un certo indirizzo **&X**.
- Ciascun elemento del vettore **X** viene riferito tramite un **indice *i*** con valori che variano da 0 a **N - 1** (elemento ***i***: **X[*i*]**)
- L'indice *i* di un elemento **X_{*i*}** determina il suo indirizzo relativo al indirizzo del vettore **X**, uguale a ***i* * byte-per-codificare-un-valore**
- L'indirizzo assoluto (nella RAM) di un dato **X_{*i*}** è uguale all'*indirizzo del vettore X + l'indirizzo relativo di X_{*i*}*.

Riferimento a dati vettoriali



- **LOAD R , Var , Ri**
LOAD R , Var , i carica nel registro-dati R il contenuto del elemento $[Ri]/i$ della variabile Var (l'indice = contenuto del registro Ri)
- **STORE R , Var , Ri**
STORE R , Var , i copia il contenuto del registro dati R nel elemento $[Ri]/i$ della variabile Var

In *microcalc*, ogni variabile potrebbe essere un vettore (massimo 20 elementi):
V: INT 10 15 20 25 30; - definisce i valori iniziali di un vettore V (i=0,1,2,3,4)
LOAD R1 V 4; - carica in R1 il 5° elemento di V
STORE R1 V 8; - imposta il valore del 9° elemento di V

ACCUMULATORE

- Se abbiamo istruzioni per sommare (moltiplicare, ecc) due numeri scalari, come possiamo applicare la stessa operazione su più numeri ?
- Soluzione: definire una variabile *accumulatore* **A** ed applicare la stessa operazione tra **A** e ogni valore in questione.
- **Algoritmo**: dati i n valori ($X_1, X_2, \dots X_n$)
 - 1) inizializzare l'accumulatore **Y**
 - 2) applicare l'operazione sull'accumulatore e ciascun valore X_i , memorizzando il risultato nell'accumulatore.
- **Inizializzazione**:

Metodo A: "Azzerare" ed accumulare

 - per *sommare / moltiplicare*, inizializzare con 0 / 1
 - per *max / min*, inizializzare con un numero piccolo / grande.
- Metodo B: assegnare all'accumulatore il primo valore X_1 , adoperando poi con tutti gli altri valori.

SOMMA VETTORIALE

V_0, V_1, \dots, V_n
 $SUM=0$
 $I=0$ (contatore)

I \geq **N** ?

falso

$SUM = SUM + V_i$

$I = I + 1$

vero

V: INT 3 10 2 -1 -4; **Vettore V**
N: INT 5; **lunghezza di V**
S: INT; **Accumulatore - Somma**

LOAD R0 0; **Indice i=0**
LOAD R1 N; **Numero elementi in V**
LOAD R2 0; **Acc=0**

Testa: COMP R0 R1; **i-N**
BRGE Post; **se i** \geq **N, fine ciclo**
LOAD R3 V R0; **V[i]**
ADD R2 R3; **Acc=Acc+V[i]**
ADD R0 1; **i=i+1;**
BRANCH **Testa;**

Post: STORE R2 S; **S=Acc**
STOP;

SOMMA VETTORIALE (2)

V_0, V_1, \dots, V_n
 $SUM = V_0$
 $I = 1$ (contatore)

I \geq N ?

falso

$SUM = SUM + V_i$

$I = I + 1$

vero

```
V: INT 3 10 2 -1 -4; Vettore V
N: INT 5; lunghezza di V
S: INT; Accumulatore - Somma
```

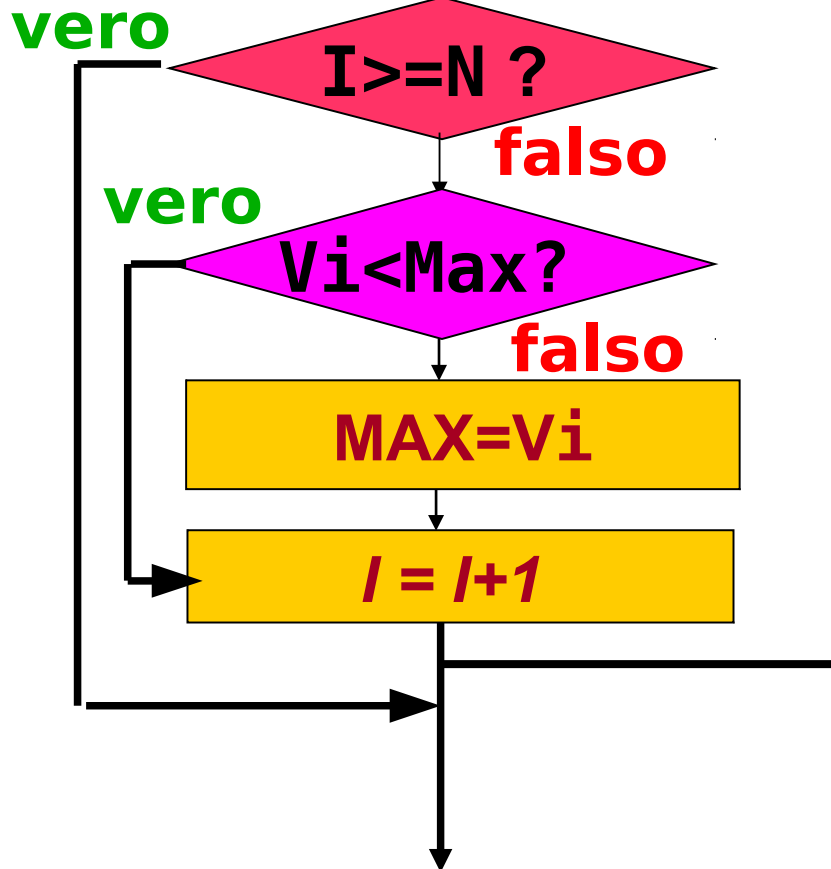
```
LOAD R0 1; Indice i=0
LOAD R1 N;
LOAD R2 V 0; Acc=V[0]
```

```
Testa: COMP R0 R1; i-N
BRGE Post; se  $i \geq N$ , fine ciclo
LOAD R3 V R0;  $V[i]$ 
ADD R2 R3;  $Acc = Acc + V[i]$ 
ADD R0 1;  $i = i + 1$ ;
BRANCH Testa;
```

```
Post: STORE R2 S;
STOP;
```

VALORE MASSIMO

V_0, V_1, \dots, V_n
 $MAX = X_0$
 $I = 1$ (contatore)



```
V: INT 3 10 2 -1 -4; Vettore V  
N: INT 5; lunghezza di V  
M: INT; Accumulatore - Max
```

```
LOAD R0 1; Indice i=0
```

```
LOAD R1 N;
```

```
LOAD R2 V 0; Max=V[0]
```

```
Testa: COMP R0 R1; i-N
```

```
BRGE Post; se i >= N, fine ciclo
```

```
LOAD R3 V R0; R2=V[i]
```

```
COMP R3 R2; Xi < Max?
```

```
BRLT Next; Yes; next
```

```
LOAD R2 R3; No: Max=Xi
```

```
Next: ADD R0 1; i=i+1
```

```
BRANCH Testa;
```

```
Post: STORE R2 S;
```

```
STOP;
```

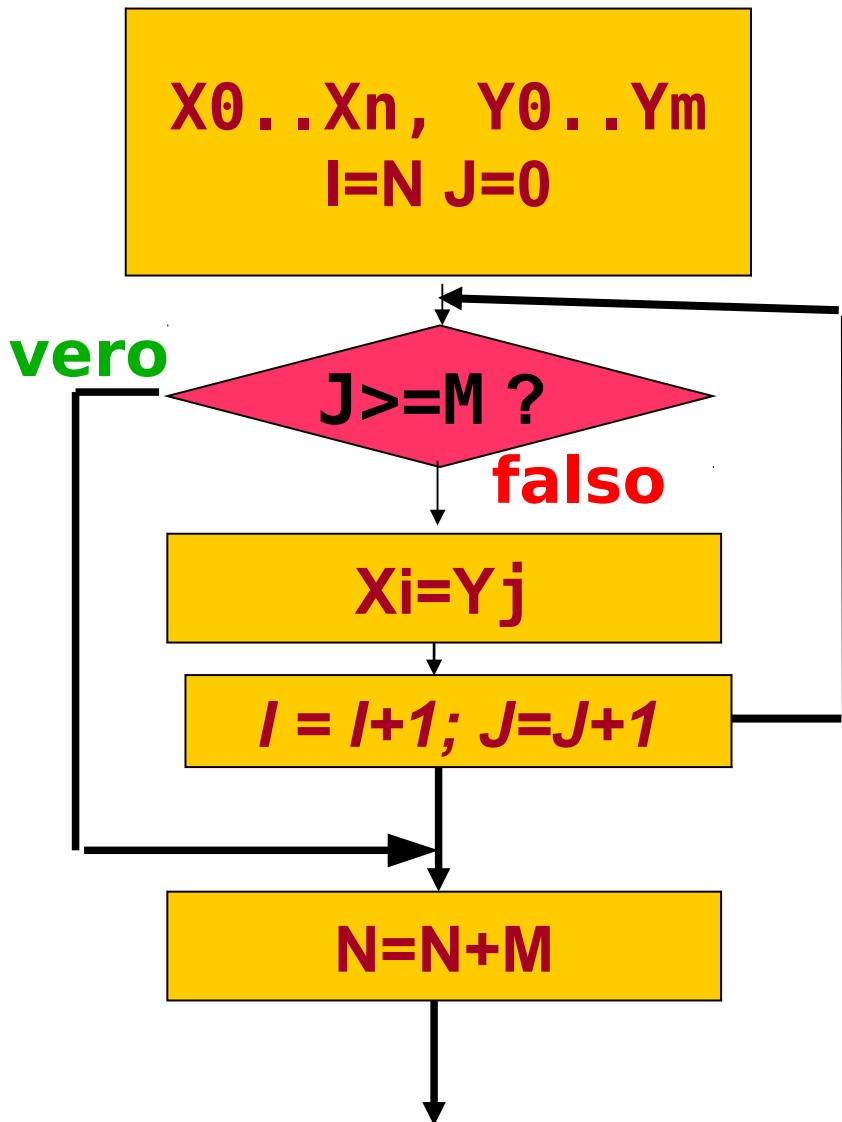
CONCATENARE DUE VETTORI

```
X: INT 3 10 2 -1; Vettore X  
N: INT 4; lunghezza di X  
Y: INT -1 -2 -3; Vettore Y  
M: INT 3; lunghezza di Y
```

```
LOAD R0 N; Indice  $i = \text{len}(X)$   
LOAD R1 0; Indice  $j = 0$   
LOAD R2 M;  $\text{len}(Y)$ 
```

```
Testa: COMP R1 R2;  $i - M$   
BRGE Post; copiati tutti  $Y[j]$   
LOAD R3 Y R1;  $Y[j]$   
STORE R3 X R0;  $X[i] = Y[j]$   
ADD R0 1;  $i = i + 1$   
ADD R1 1;  $j = j + 1$   
BRANCH Testa;
```

```
Post: STORE R0 N;  $\text{len}(X) = N + M$   
STOP;
```



COVARIANZA $E(XY) - E(X)E(Y)$

X_0, X_1, \dots, X_n

Y_0, Y_1, \dots, Y_n

$I=1$ (contatore)

$E_x=0, E_y=0; E_{xy}=0$

vero

$I \geq N ?$

falso

$E_x = E_x + X_i$
 $E_y = E_y + Y_i$
 $E_{xy} = E_{xy} + X_i Y_i$

$I = I + 1$

```
X: FLOAT 3 10 2 -1 -4;
```

```
Y: FLOAT 1 2 -2 -3 -8;
```

```
N: FLOAT 5; lunghezza di X,Y
```

```
Cov: FLOAT; Covarianza
```

```
LOAD R0 0; Indice i=0
```

```
LOAD R1 N;
```

```
LOAD R2 0;  $E_x=0$ 
```

```
LOAD R3 0;  $E_y=0$ 
```

```
LOAD R4 0;  $E_{xy}=0$ 
```

```
Testa: COMP R0 R1;  $i-N$ 
```

```
BRGE Post; se  $i \geq N$ , fine ciclo
```

```
LOAD R5 X R0;  $X_i$ 
```

```
LOAD R6 Y R0;  $Y_i$ 
```

```
LOAD R7 R5;
```

```
MUL R7 R6;  $X_i * Y_i$ 
```

```
ADD R2 R5;  $E_x = E_x + X_i$ 
```

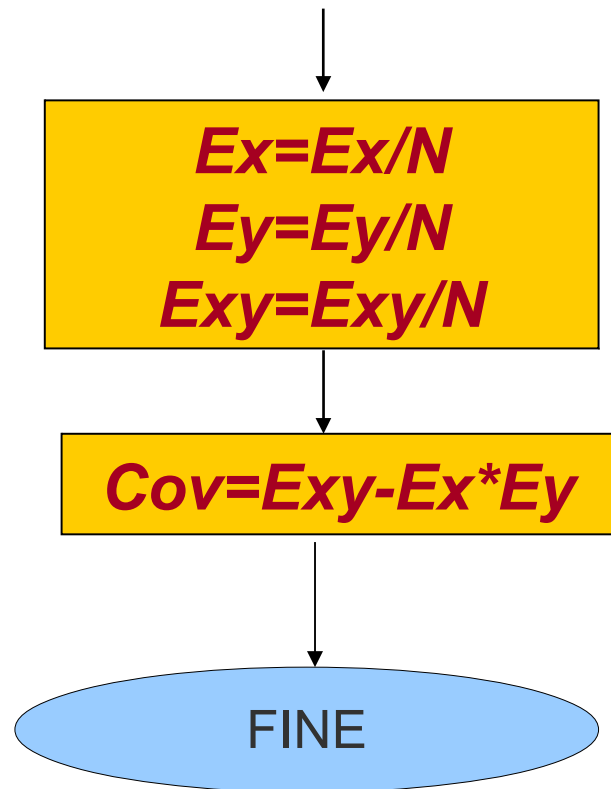
```
ADD R3 R6;  $E_y = E_y + Y_i$ 
```

```
ADD R4 R7;  $E_{xy} = E_{xy} + X_i Y_i$ 
```

```
ADD R0 1;  $i = i + 1$ ;
```

```
BRANCH Testa;
```

COVARIANZA (2) $E(XY) - E(X)E(Y)$



```
Post: DIV R2 R1; Ex=Ex/N
      DIV R3 R1; Ey=Ey/N
      DIV R4 R1; Exy=Exy/N

      MUL R2 R3; Ex*Ey
      SUB R4 R2; Exy-Ex*Ey
      STORE R4 Cov; Cov=Exy-Ex*Ey

      STOP;
```

Ulteriori esercizi

- 1) Ordinare tre variabili X, Y, Z
- 2) Se N è pari, calcolare la progressione $2+4+..N$;
se N è dispari, calcolare $1+3+5+...N$
- 3) Dato un vettore X , creare un nuovo vettore Y con elementi $Y[i]$ uguali a $-X[i]$.
- 4) Dati due vettori X e Y , concatenare il vettore Y al vettore X .

Esercizio per eccellenza

- Dato il vettore X , creare un altro vettore Y che contiene i valori del vettore X in ordine crescente.