Architettura Elaboratori Elettronici

ESERCITAZIONI

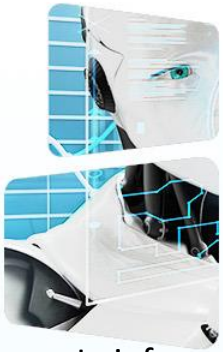
STRUTTURE DATI

Franco Liberati

liberati@di.uniroma1.it

Argomenti

- Strutture dati statiche
  - Vettore
  - Matrice
- Strutture dati dinamiche
  - Lista
  - Code
  - Stack



Struttura Dati

Definizione

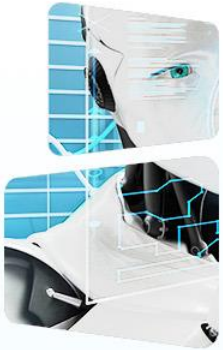
In informatica per **struttura dati** si intende un modo specifico di organizzare e memorizzare i dati (elementari) in modo che possano essere utilizzati in modo efficiente

In altre parole una struttura dati è una definizione di dati e di operazioni che possono essere eseguite su di essi. In sostanza, una struttura dati fornisce un modo per organizzare e manipolare i dati in modo efficiente e significativo all'interno di un programma informatico.

I dati sfruttati nelle strutture dati sono quelli gestiti direttamente dall’elaboratore: byte, halfword, word, float, double, ascii

Le strutture dati sono progettate per ottimizzare le operazioni di accesso, modifica e ricerca dei dati

Tra le strutture dati più note ci sono: vettori, liste, code, pile, alberi, grafi.



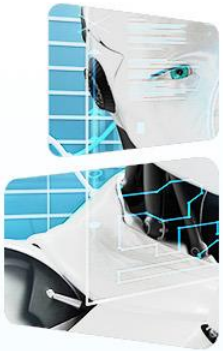
Struttura Dati

Classificazione

Le strutture dati possono essere **statiche** o **dinamiche**

- Nelle strutture dati statiche il numero dei dati è prestabilita e non muta (vettore o matrice)
- Nelle strutture dati dinamiche il numero di dati cambia nel corso dell'esecuzione del programma (pila, coda, lista, grafo). In questo ultimo caso un campo dei dati può essere usato per contenere l'indirizzo di un'ulteriore elemento della struttura dati (si dice che punta o che è un puntatore a quest'ultima)

VETTORE



VETTORE

Generalità

Un **vettore** (o **array**) è un insieme ordinato di elementi tutti dello stesso tipo disposti sequenzialmente in memoria centrale

In informatica i vettori possono essere utilizzati in diversi contesti e offrono molteplici vantaggi:

- **Memorizzazione sequenziale**: i vettori memorizzano elementi in modo sequenziale in memoria, consentendo un accesso rapido agli elementi tramite un indice
- **Accesso casuale**: Gli elementi di un vettore possono essere accessibili in modo casuale tramite l'indice corrispondente, il che rende l'accesso ai dati efficiente
- **Implementazione di strutture dati complesse**: i vettori sono spesso utilizzati come base per implementare altre strutture dati complesse come liste, code e pile.
- **Algoritmi di elaborazione dati**: Molte operazioni di elaborazione dati, come ordinamento, ricerca e manipolazione, possono essere implementate efficientemente utilizzando vettori

VETTORE

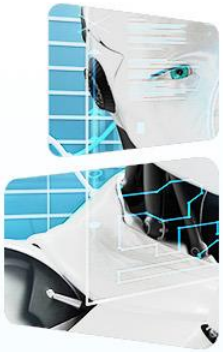
Definizione fisica

Un vettore è definito esplicitando il numero di elementi che lo compongono (**lunghezza del vettore**)

L'accesso ad un elemento del vettore è consentito specificando un **indice**

Il **primo elemento** del vettore ha **indice 0**

Indice	Indirizzo in memoria	Valore in memoria
0	**100**	**12**
1	**104**	**34**
2	**108**	**-121**
3	**112**	**0**
4	**116**	**54**



VETTORE

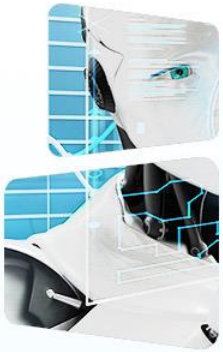
Definizione in un linguaggio ad alto livello

Un **vettore è definito**, in un linguaggio ad alto livello, dalla specifica del tipo, dall'identificatore e dalla sua lunghezza (cioè del numero di elementi)

È possibile inizializzare un elemento di un vettore specificando l'**identificativo** e l'**indice** (nel linguaggio C si usano le parentesi quadre) e il **valore**

Per prelevare un elemento è sufficiente utilizzare l'**identificativo** e l'**indice** (e una variabile per la memorizzazione del dato)

Dichiarazione	**int array[5];**	**Crea un vettore di interi in memoria (cinque elementi consecutivi)**
Inizializzazione	**array[0]=45;**	**Inserisce nella prima locazione di memoria assegnata all'array il valore 45**
Recupero dati	**int temp=array[0]**	**Nella variabile temp si trova il valore relativo alla prima locazione di memoria associata all'array (ovvero temp=45)**



VETTORE

Manipolazione usando un linguaggio ad alto livello

Effettuare la somma degli elementi di due vettori (di lunghezza 3) a medesima posizione

ESEMPIO

INPUT:

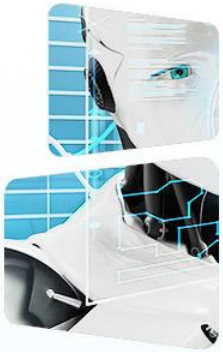
V1=(13,24,35)

V2=(47,16,-25)

OUTPUT:

V=(60,40,10)

```
#include <stdio.h>
int main()
{
   const int dim_vett=3;
   int i, a,b,addition;
   int v1[dim_vett]={13,24,35};
   int v2[dim_vett]={47,16,-25};
   int v3[dim_vett];
   for (i=0;i<dim_vett;i=i+1){
      a=v1[i];
      b=v2[i];
      addition=a+b;
      v3[i]=addition;
   }

   for (i=0;i<dim_vett;i=i+1){
      printf("%d\t", v3[i]);
   }
return 0;
}
```

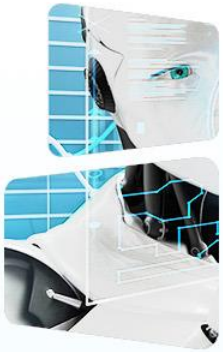


VETTORE IN MARS

Generalità

- Un **vettore** è un sequenza di dati omogenei *(dello stesso tipo)* disposti sequenzialmente in memoria
  - I dati o *elementi* di un array hanno tutti la stessa dimensione di tipo (*esize* o *dimension*) correlata al tipo elementare: **word** sono 4byte, **half** sono 2byte, **byte** sono 8bit per elemento
  - La posizione (*indice*) identifica i singoli elementi

Indice	Indirizzo in memoria	Valore in memoria
0	**100**	**12**
1	**104**	**34**
2	**108**	**-121**
3	**112**	**0**
4	**116**	**54**



VETTORE IN MARS

Definizione

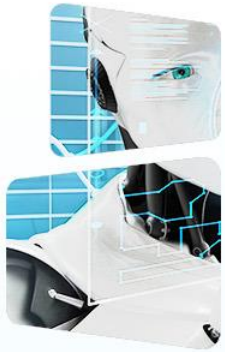
- In MARS un vettore si definisce specificando il tipo e interponendo delle virgole tra gli elementi

- Essendo una struttura statica in MARS è doveroso riportare anche il numero di elementi che lo costituiscono (**lunghezza del vettore**)

Esempio:

array8bit: .byte 12,34,-121,0,54
lunghezzaarray8bit: .byte 5

Indice	Posizione	Valore in memoria
0	100	**12**
1	101	**34**
2	102	**-121**
3	103	**0**
4	104	**54**



VETTORE IN MARS

Definizione

Esempio:

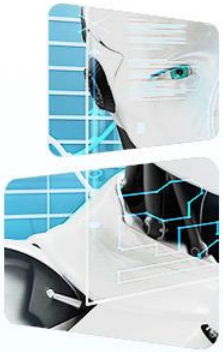
array16bit: .half 12000,-2000,-15,78
lunghezzaarray16bit: .byte 4

Indice	Posizione	Valore in memoria
0	200	**12000**
1	202	**-2000**
2	204	**-15**
3	206	**78**

Esempio:

array32bit: .word 67067,23400000,11,-785
lunghezzaarray32bit: .byte 4

Indice	Posizione	Valore in memoria
0	300	**67067**
1	304	**23400000**
2	308	**11**
3	312	**-785**



VETTORE IN MARS

Inizializzazione

- È possibile inizializzare un vettore ad un valore specifico definendo il valore e la sua lunghezza (numero di elementi) con una sintassi del tipo

etichetta: .tipo valore_di_inizializzazione:numero_elementi

- Esempio:

vettore: .word 2:10

Posizione	Valore in memoria
0	2
4	2
8	2
12	2
16	2
20	2
24	2
28	2
32	2
36	2

VETTORE IN MARS

Individuazione di un elemento

- **L'indirizzo dell'elemento** è ottenuto da:

base + index* dimension

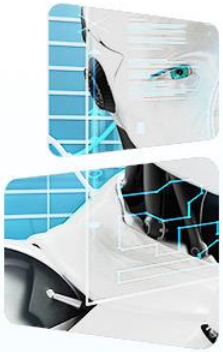
dove

base **è l'indirizzo di memoria dove inizia il vettore**

index **è l'incide dell'elemento che si vuole manipolare (inizializzare o prelevare)**

dimension **è la dimensione degli elementi (dipende dal tipo elementare)**

Posizione	Valore in memoria
0	2
4	2
8	2
12	2
16	2
20	2
24	2
28	2
32	2
36	2



VETTORE IN MARS

Individuazione di un elemento

Esempio:

array32bit: .word 123,56,44,22,-33,67
lunghezzaarray32bit: .byte 6

- L'indirizzo del quarto elemento è ottenuto da:

base* + *index* * *dimension

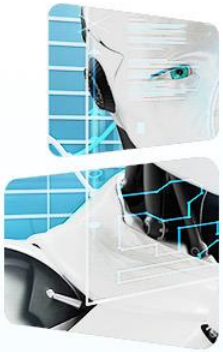
dove
base **è 100**

index **è 3**

dimension **è 4**

Cioè locazione di memoria: **112**

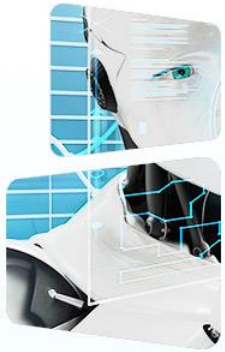
Indice		Valore in memoria
0	Array >100	123
1	104	56
2	108	44
3	112	22
4	116	-33
5	120	67



VETTORE IN MARS

Manipolazione dei dati

Per accedere agli elementi di un vettore in lettura e in scrittura si può ricorrere a due modalità



VETTORE IN MARS

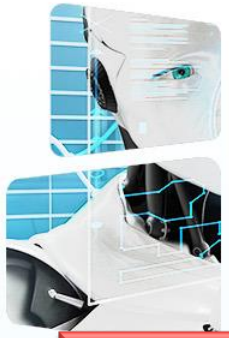
Manipolazione dei dati: a registro con spiazzamento simbolico

❑Per accedere agli elementi di un vettore in lettura e in scrittura si può ricorrere al **modo di indirizzamento a registro con spiazzamento simbolico**

vettore($a0)

Dove $a0 *deve contenere la posizione dell'elemento moltiplicato per la sua dimensione (4 in caso di word, 2 per le halfword, 1 per il byte)*

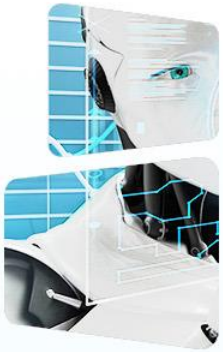
```
li $t1,0        #indice relativo al primo elemento
mul $a0,$t1,4 #indice*dimensione (posizione relativa)
lw $t0,array($a0) # copia/prelievo del primo elemento
                  #il calcolo su array($a0) riporta
                  #la posizione assoluta

li $t1,3        #indice relativo al quarto elemento
mul $a0,$t1,4 #posizione relativa del quarto elemento
lw $t3,array($a0) #prelievo del quarto elemento

.data
array: .word 45,100,-23,56
lunghezzaarray: .byte 4
```



VETTORE IN MARS

Manipolazione dei dati

Stampa di un vettore lungo 5 elementi di dimensione 16bit

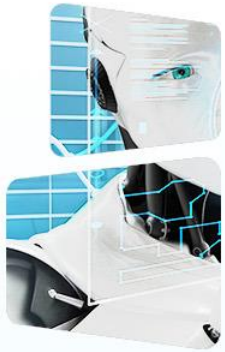


VETTORE IN MARS

Manipolazione dei dati

```
.text
.globl main
main:
        # STAMPA DI UN VETTORE
        lw $t1,nunelem        # $t1 numero elementi del vettore
        li $t2,0  # $t2 indice
loop:
        mul $t3,$t2,2         # indice*dimensione (moltiplicare x 2 perché halfword)
        lh $t4,array($t3)     # copia dell'i-esimo elemento nel registro $t4 cioè $t4=v[i]
        move $a0,$t4          #sposto l'elemento per effettuarne la stampa
        li $v0,1               #stampo i-esimo elemento
        syscall
        la $a0,tabulato       # Stampo un tabulato (per una maggiore leggibilità)
        li $v0,4
        syscall
        add $t2,$t2,1         # incremento indice
        blt $t2,$t1,loop      #confronto per determinare la fine del vettore
li $v0,10
syscall
```

```
.data
array: .half 12,43,23,54,77
nunelem: .word 5
tabulato: .asciiz "\t"
```



VETTORE IN MARS

Manipolazione dei dati: a registro con spiazzamento

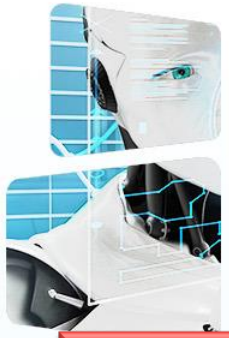
- Un'altra possibilità è quella di utilizzare come modo di indirizzamento per accedere agli elementi **a registro con spiazzamento**

($a0)

dove $a0 deve contenere un indirizzo costituito da

posizione_vettore+ (posizione_elemento)*dimensione

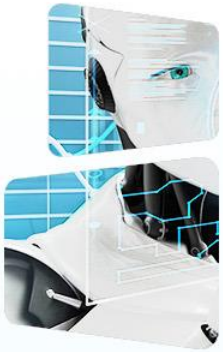
```
li $t1,0          # indice relativo del primo elemento
mul $t1,$t1,4     # indice*dimensione (posizione relativa)
la $a0,array      # indirizzo di inizio del vettore
add $a0,$a0,$t1
lw $t0,($a0)      # copia del primo elemento
li $t1,3          # indice relativo del IV elemento
mul $t1,$t1,4     # indice*dimensione (posizione relativa)
add $a0,$a0,$t1   # aggiornamento  posizione fisica del IV
                  #elemento
lw $t1, ($a0)     # copia/prelievo del quarto elemento

.data
array: .word 45,100,-23,56
lunghezzaarray: .byte 4
```



VETTORE IN MARS

Manipolazione dei dati

Stampa di un vettore lungo 5 elementi di dimensione 16bit



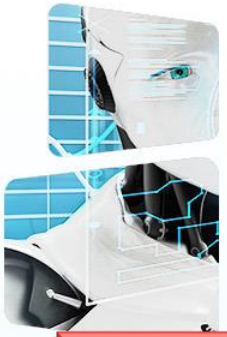
VETTORE IN MARS

Manipolazione dei dati

```
.text
.global main
main:
        la $t0,array                    # locazione di inizio del vettore
        lw $t1,nunelem                  # $t1 numero elementi del vettore
        li $t2,0                # $t2 indice
loop:
        mul $t3,$t2,4                   # indice * dimensione
        add $t3,$t0,$t3                 #posizione+(indice*dimensione)
        lh $t4,($t3)                    # copia dell'i-esimo elemento in $t4
        move $a0,$t4                    #... Elaboro elemento (es.:stampa)
        li $v0,1
        syscall
        la $a0,tabulato         # Stampo un tabulato per una maggiore leggibilità
        li $v0,4
        syscall
        addu $t2,$t2,1                  # incremento indice
        blt $t2,$t1,loop
li $v0,10
syscall
```

Si guadagna in efficienza

```
.data
array: .half 12,43,23,54,77
nunelem: .word 5
tabulato: .asciiz "\t"
```



ESERCIZIO

Definire due vettori di 5 elementi x di valore $-2^{16}<x_i<2^{15}-1$ e eseguire il **prodotto scalare (⊥)**

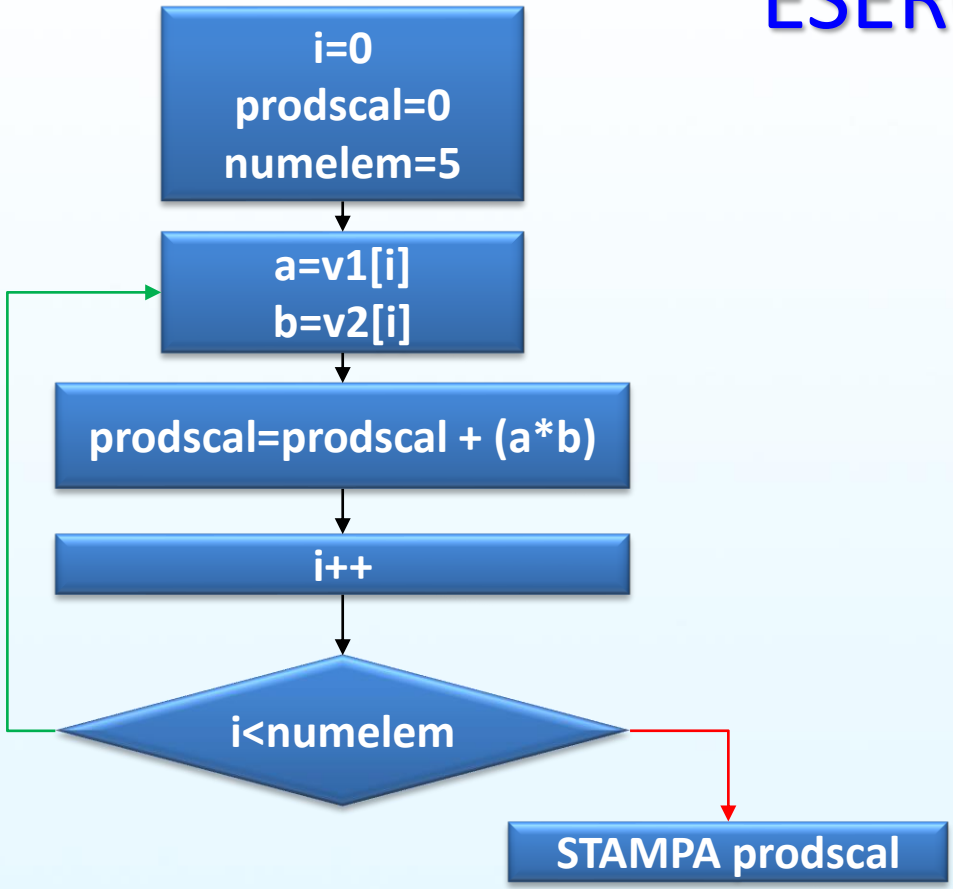
NB:

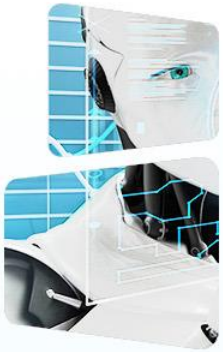
v1=(3,5,8,10,1) e v2=(1,2,3,0,13)

$v1 \perp v2 = (3\cdot1)+(5\cdot2)+(8\cdot3)+(10\cdot0)+(1\cdot13)=50$

Prodotto scalare

ESERCIZIO

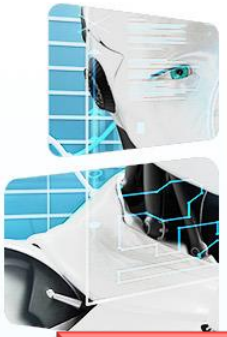




ESERCIZIO

```
.text
.globl main
main:
        xor $t0,$t0,$t0  #inizializzazione a zero del registro contenete l'indice del vettore
        li $t5,0         #inizializzatore del registro che contiene il prodotto scalare
        lw $t9,numelem           #lunghezza del vettore
ciclo:
        mul $t1,$t0,2             #spiazzamento all'i-esimo elemento indice*dimensione
        lh $t2,v1($t1)                    #v1[i]
        lh $t3,v2($t1)                    #v2[i]
        mul $t4,$t2,$t3           #v1[i]*v2[i]
        add $t5,$t5,$t4           #prodotto scalare parziale
        add $t0,$t0,1                     #incremento indice
        bne $t0,$t9,ciclo         #confronto fine vettore

        move $a0,$t5              #stampa del prodotto scalare
        li $v0,1
        syscall
```

```
.data
v1: .half 3,5,8,10,1
v2: .half 1,2,3,0,13
numelem:.word 5
```



ESERCIZIO II

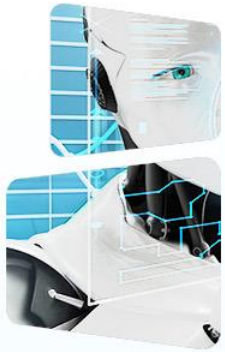
- Definire un vettore v1 di 6 elementi x di valore $-2^{32}<x_i<2^{31}-1$ e memorizzarlo in un nuovo vettore includendo solo gli elementi multipli di 5. Stampare v2

INPUT

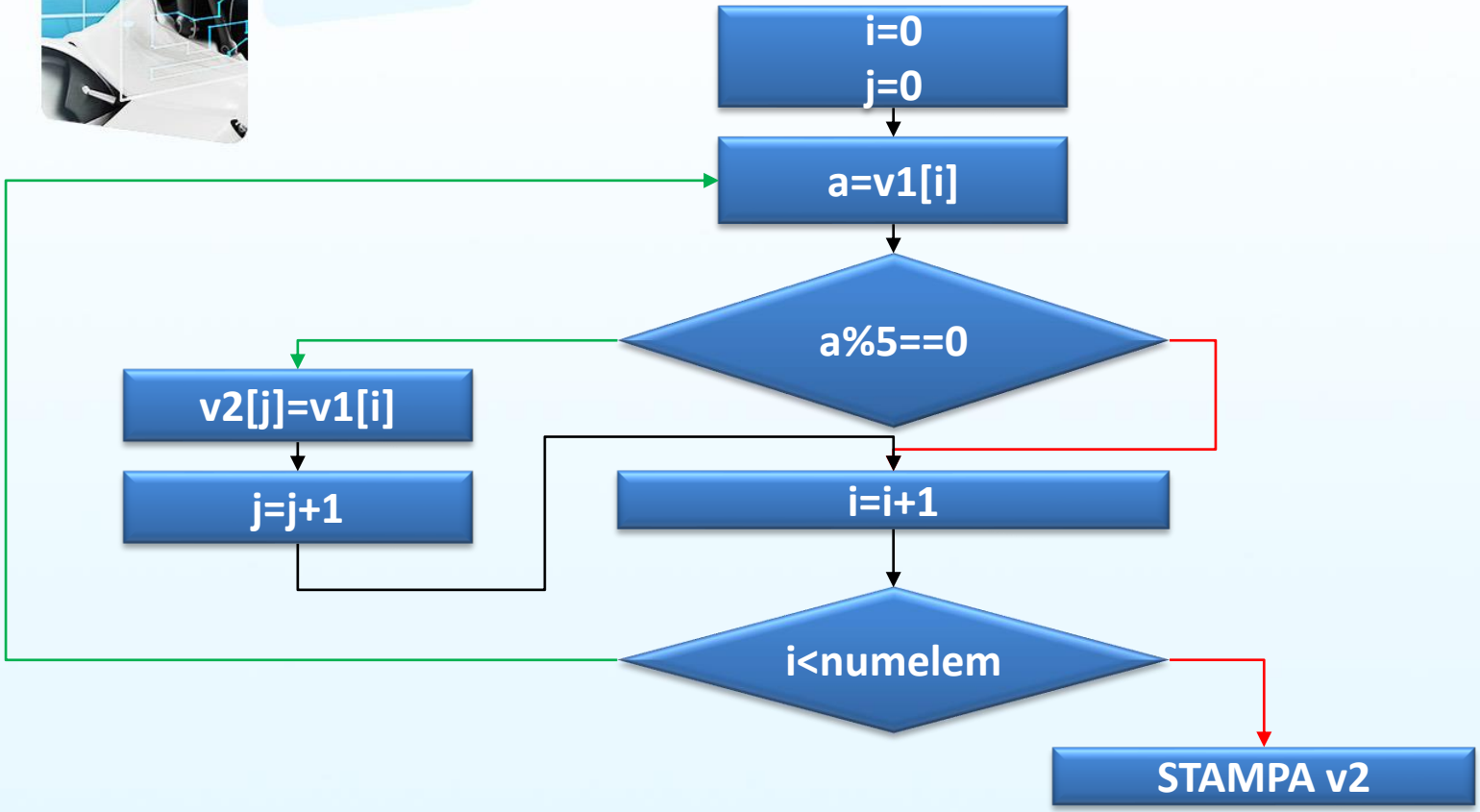
v1=(3,5,8,10,4,20)

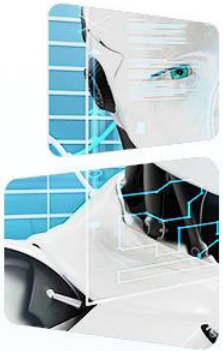
OUTPUT

v2= (5,10,20)



ESERCIZIO II





ESERCIZIO II

```
.text
.globl main
main:
        la $t0,v1        # indirizzo al primo elemento di v1
        la $t1,v2        # indirizzo al primo elemento di v2
        lw $t2,numelem   # $t2 numero elementi del vettore
        mul $t9,$t2,4    # spiazzamento dell'ultimo elemento di v1
        add $t9,$t9,$t0  # indirizzamento dell'ultimo elemento di v1

loop:   lw $t3,($t0)     #copia dell'i-esimo elemento di v1 nel registro $t3
        rem $t4,$t3,5            # vedo se v1[i] è multiplo di 5
        bnez $t4,salta
        sw $t3,($t1)             #memorizzo v2[j]=v1[i]
        addu $t1,$t1,4           # incremento indice v2

salta:  addu $t0,$t0,4           # incremento indice v1
        bgt  $t9,$t0,loop        #ciclo se non ho letto tutto  v1
        move $t8,$t1             # dimensione v2
        la $t1,v2                # lettura posizione inziale v2

stampa: lw $a0,($t1)
        li $v0,1
        syscall
        la $a0,separatore
        li $v0,4
        syscall
        addu $t1,$t1,4
        blt $t1,$t8, stampa
        li $v0,10
        syscall

.data
v1: .word 3,5,8,10,1,7
v2:.space 24
numelem:.word 6
separatore:.asciiz "\t"
```

Si calcola
l'indirizzo relativo all'inizio del vettore
per poi svolgere il salto