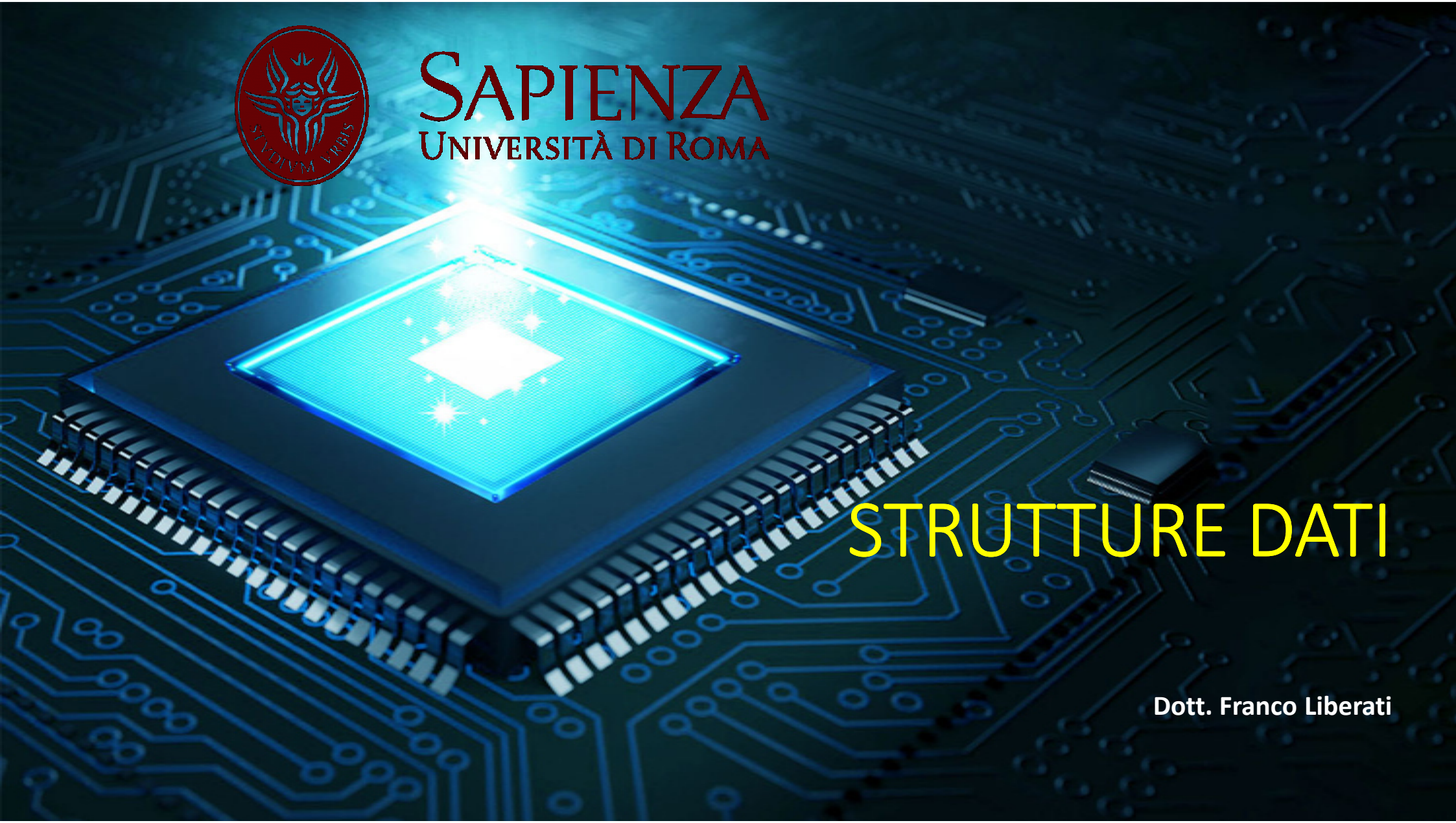
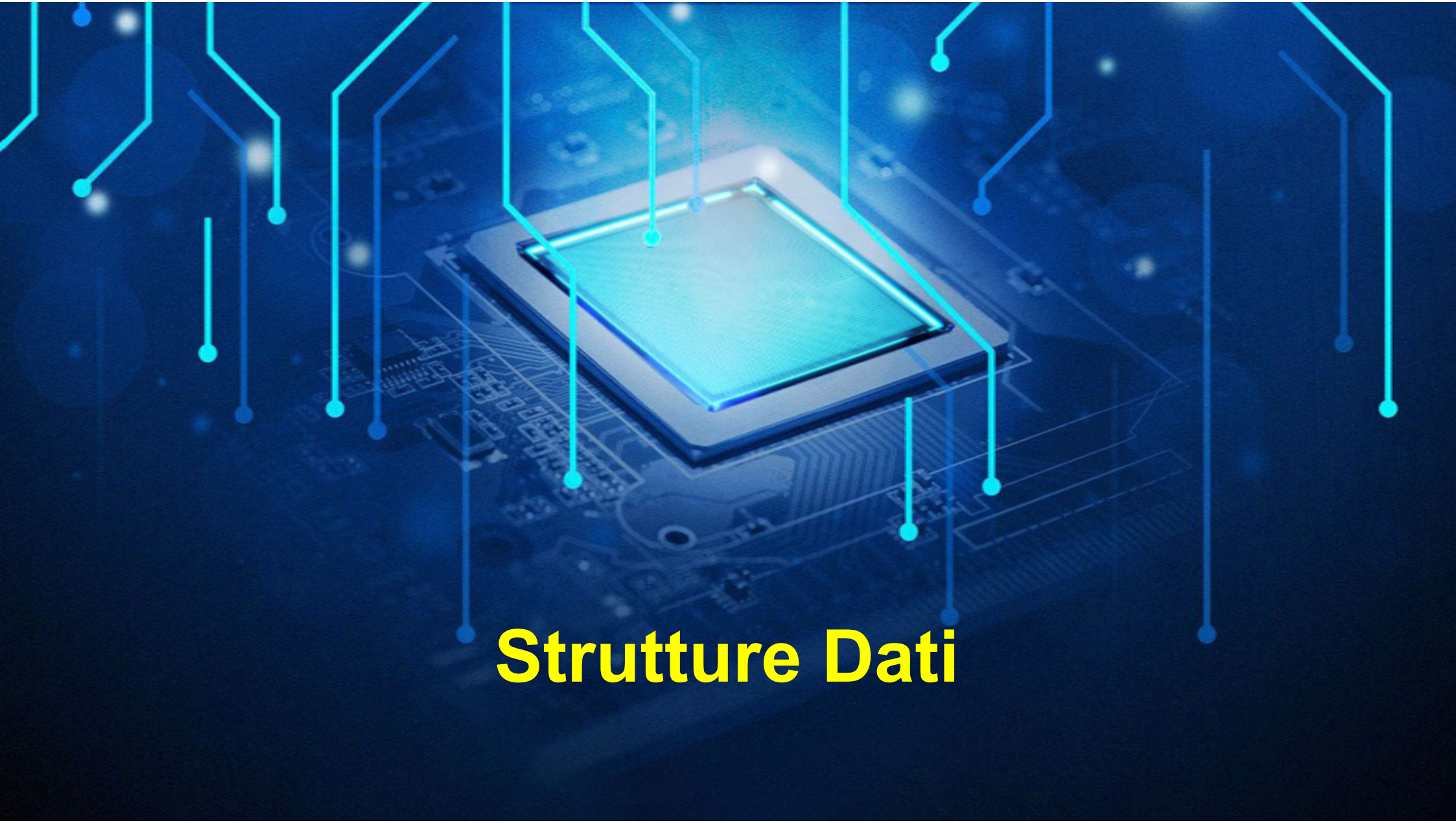


STRUTTURE DATI

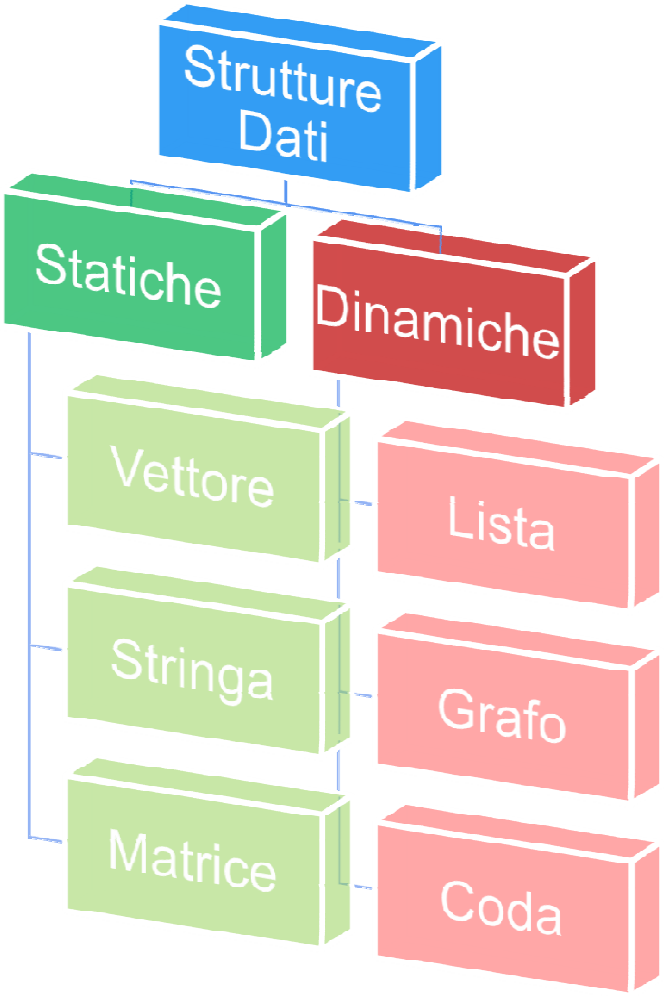
Dott. Franco Liberati


Argomenti
01
Strutture dati
02
Vettore (array)
03
Stringa
04
Matrice (array multidimensionali)


Strutture Dati

Strutture Dati

Una struttura dati definisce il modo in cui sono organizzate delle entità di informazioni. Le entità possono essere **statiche**, se rimangono della stessa dimensione per tutta l'esecuzione (occupano una area di memoria definita e immutabile); o **dinamiche**, quando cambiano la loro dimensione, con l'aggiunta o la sottrazione di elementi, in accordo con l'evoluzione del programma




Vettore

Vettore

Generalità

Un vettore è un insieme di operandi omogenei (dello stesso tipo nei linguaggi ad alto livello, della stessa dimensione dell'operando nei linguaggi assemblativi) **e statico** (occupa un insieme di celle di memoria che non mutano posizione durante l'esecuzione del programma)

- 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 e conoscendo la dimensione dell'elemento
- Il primo elemento del vettore ha indice 0

Indice	Elemento in memoria
0	**12**
1	**34**
2	**-121**
3	**0**
4	**54**

Vettore

Definizione

In MIPS un vettore si definisce specificando la dimensione dell'operando e interponendo delle virgole tra gli elementi
La **posizione effettiva** in memoria, pertanto, dipende dalla direttiva di dimensione
Quando si definisce un vettore è lecito riportare anche il numero di elementi che lo costituiscono (**lunghezza del vettore**)

array8bit: .byte 12,34,-121,0,54
Lunghezza_array8bit: .byte 5

Posizione in memoria	Elemento in memoria
0	**12**
1	**34**
2	**-121**
3	**0**
4	**54**

array16bit: .half 12000,-2000,-15,78
Lunghezza_array16bit: .byte 4

Posizione in memoria	Elemento in memoria
0	**12000**
2	**-2000**
4	**-15**
6	**78**

array32bit: .word 67067,23400000,11,-785,0
Lunghezza_array32bit: .byte 5

Posizione in memoria	Elemento in memoria
0	**67067**
4	**23400000**
8	**11**
12	**-785**
16	**0**

Vettore

Accesso agli elementi

Gli **elementi** di un vettore hanno tutti la stessa dimensione (***dimension*** o ***element size***)

I singoli elementi sono identificati dalla **posizione** (***index***)

L'**indirizzo assoluto** dell'elemento i-esimo è ottenuto da:

base* + *index* dimension

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

Indice	*Posizione in memoria*	Elemento in memoria
0	*100*	**67067**
1	*104*	**23400000**
2	*108*	**11**
3	*112*	**-785**
4	*116*	**0**
5	*120*	**1205**
6	*124*	**1807**

Terzo elemento: 100+2*4=108
Settimo elemento: 100+6*4=124

Vettore

Modi di indirizzamento per l'accesso agli elementi

Per accedere ad un dato di un vettore si può ricorrere al modo di indirizzamento **differito a registro con spiazzamento simbolico**
Dove il simbolo è l'etichetta che definisce il vettore e il registro contiene l'indice dell'elemento (indice relativo) da elaborare moltiplicato per la base (indice assoluto)

```
            #Scansione di un vettore
            …
            li $t0,0    #indice relativo
            li $t1,4    #dimensione
            lb $t2, lun_vett #num_elementi
ciclo:
            mul $a0,$t0,$t1              #indice assoluto
            lw $t9, vettore($a0)         #prelievo elemento
            … #lavoro con il dato

            add $t0,$t0,1       #incremento indice relativo
            bgt $t2,$t0, ciclo
            ….

            .data
vettore:.word 567,5543,10000,-56
lun_vett: .byte 4
```

Vettore

Esempio

```
    .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)   # carico l'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,spazio       #Stampa di uno spazio (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
spazio: .asciiz " "
```

Vettore

Modi di indirizzamento per l'accesso agli elementi (alternativa)

In alternativa si può accedere ad un dato di un vettore ricorrendo al modo di indirizzamento **differito a registro**

Dove il registro contiene la posizione iniziale del vettore e il contenuto si ottiene manipolando il suo valore in accordo alla posizione e alla dimensione

```
            #Scansione di un vettore
            …
            la $a0,vettore #indirizzo assoluto inizio del vettore
             #in $t1calcolo indirizzo ultimo elemento
            #del vettore (indirizzo assoluto)
            li $t1,$lun_vett #
            mul $t1,$t1,4     #
            add $t1,$t1,$a0 #
ciclo:
            lw $t0, ($a0)   #prelievo dell'elemento
            … #lavoro con il dato
            add $a0,$a0,4 #incremento all'elem. succ.
            blt $a0,$t1, ciclo
            …
            .data
vettore:.word 567,5543,10000,-56
lun_vett: .byte 4
```

Vettore

Esempio

```
    .text
    .globl main
main:                       # STAMPA DI UN VETTORE
        lw $t1,nunelem      # $t1 numero elementi del vettore
        la $t3,array        # $t2 indice
        mul $t9,$t1,4        #posizione relativa dell'ultimo elemento
        add $t9,$t9,$t3      #posizione relativa dell'ultimo elemento indirizzo  assoluto
loop:
        lw $t4,($t3)        # carico l'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,spazio       #Stampa di uno spazio (per una maggiore leggibilità)
        li $v0,4
        syscall
        add $t3,$t3,4       #incremento indice
        blt $t3,$t9,loop    #confronto per determinare la fine del vettore
    li $v0,10
    syscall
```

```
.data
array: .word 12,43,23,54,77
nunelem: .word 5
spazio: .asciiz " "
```

Vettore

Esercizio

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

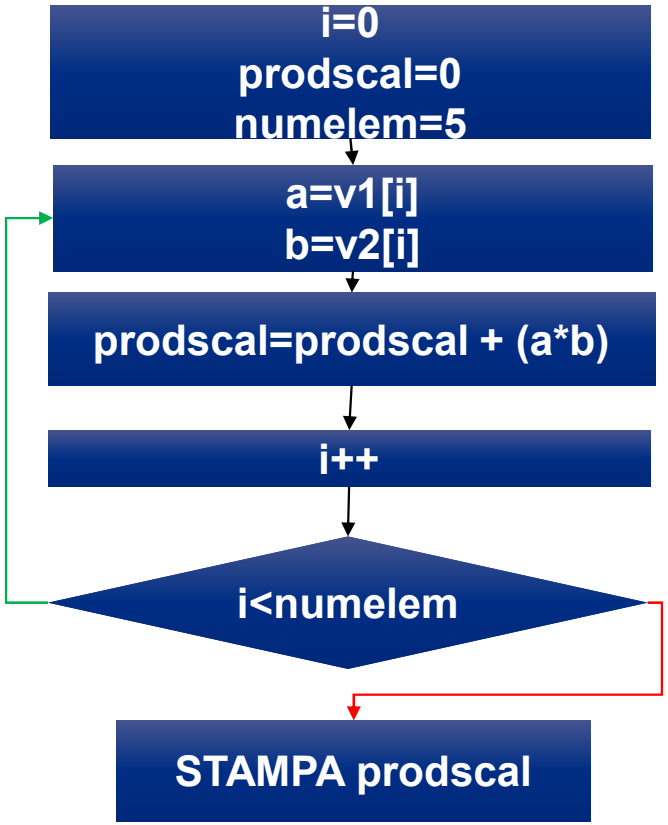
Esempio:

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

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

Vettore

Esercizio (prodotto scalare)




```
                .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 numero di elementi
                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
blt $t0,$t9,ciclo       #confronto fine vettore

                move $a0,$t5            #stampa del prodotto scalare
                li $v0,1
                syscall
                li $v0,10               #stampa terminazione
                syscall

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

Vettore

Esercizio

Definire due vettori di 5 elementi x di valore $-2^{32}<x_i<2^{32}-1$ e memorizzare il **prodotto vettoriale**

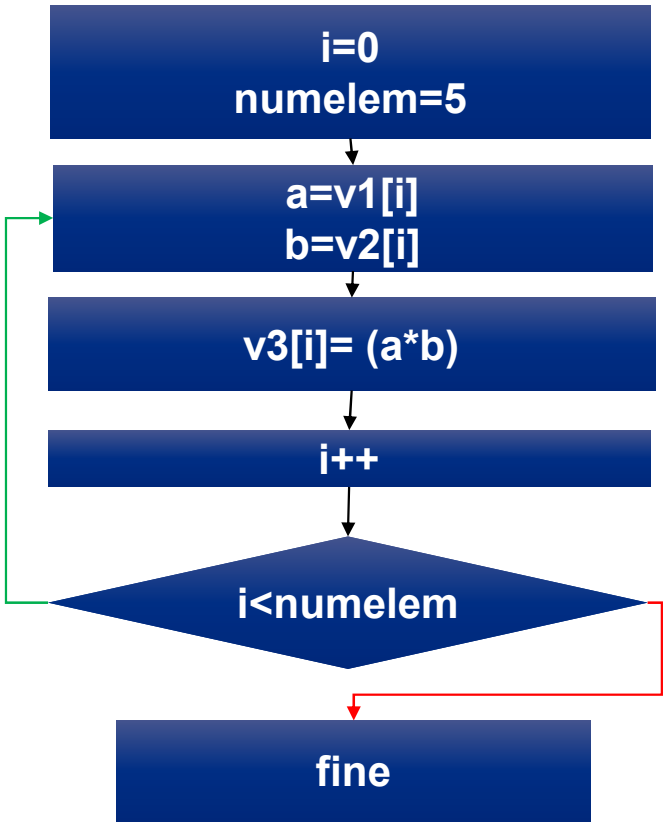
Esempio:

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

v3= (3·1,5·2,8·3,10·0,1·13)=(3,10,24,0,13)

Vettore

Esercizio (prodotto vettoriale)




```
                .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 numero di elementi
                lw $t9,numelem          #lunghezza del vettore
ciclo:
                mul $t1,$t0,4           #spiazzamento all'i-esimo elemento indice*dimensione
                lw $t2,v1($t1)          #v1[i]
                lw $t3,v2($t1)          #v2[i]
                mul $t4,$t2,$t3         #v3[i]=v1[i]*v2[i]
                sw $t4,v3($t1)          #archiviazione v[3]
                add $t0,$t0,1           #incremento indice
                bne $t0,$t9,ciclo       #confronto fine vettore

                li $v0,10               #stampa terminazione
                syscall

                .data
v1: .word 3,5,8,10,1
v2: .word 1,2,3,0,13
v3: .word 0,0,0,0,0     #inizializza un vettore di 5 elementi impostati a 0 oppure v3: .word 0:5
numelem:.word 5         #Numero di elementi dei vettore
```

Vettore

Inizializzazione di un vettore e definizione di un vettore vuoto

In MARS/MIPS è possibile inizializzare un vettore ad un valore costante per ogni elemento usando la sintassi

etichetta: .dimensione valore_costante:numero_di elementi

Es.:

array:.word 0:10 #crea un vettore di 10 elementi word inizializzati a 0
#equivale a scrivere array:.word 0,0,0,0,0,0,0,0,0,0

In alternativa si può riservare uno spazio vuoto di byte grazie alla direttiva **.space**

ES.:

vettore: .space 40 #è lo spazio usato per gestire un vettore di 10 elementi word
è lo spazio usato per gestire un vettore di 20 elementi halfword
#è lo spazio usato per gestire un vettore di 40 elementi byte

Vettore

Esercizio

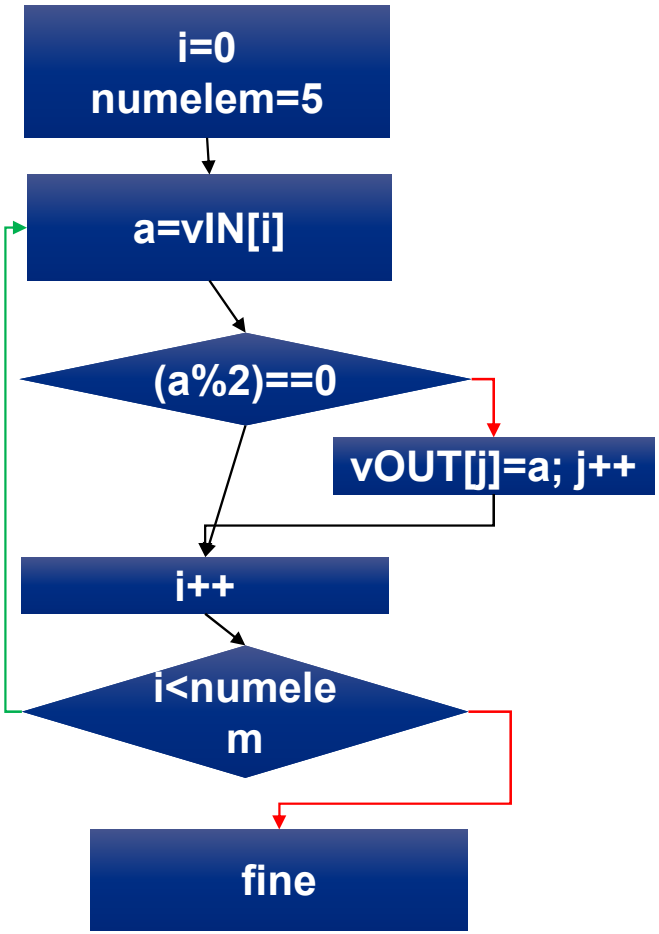
Definire due vettori di 5 elementi x di valore $-2^{32}<x_i<2^{32}-1$ e memorizzare in un nuovo vettore solamente gli elementi dispari

Esempio:

vIN=(3,5,8,10,1)
vOUT= (3,5,1)

Vettore

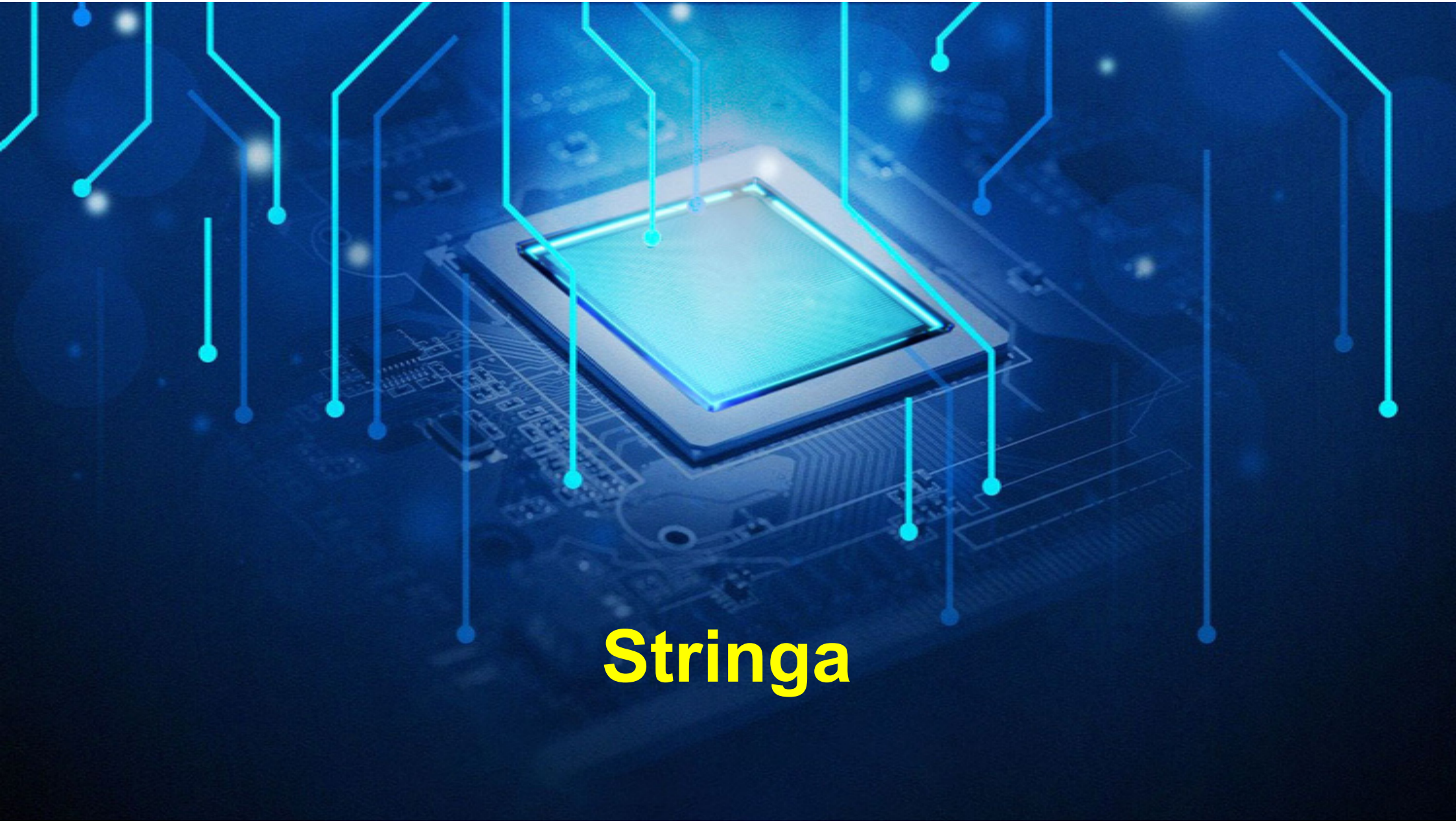
Esercizio (elementi dispari)




```
                .text
                .globl main
main:
                li $t0,0            #inizializzatore indice vIN
                li $t2,0            #inizializzatore indice OUT
                lw $t9,numelem      #lunghezza del vettore
ciclo:
                mul $t1,$t0,4       #spiazzamento all'i-esimo elemento vIN indice*dimensione
                mul $t3,$t2,4       #spiazzamento all'i-esimo elemento vOUT
                lw $t4,vIN($t1)     #lettura vIN[i]
                rem $t5,$t4,2       #Calcolo per vedere s el'elemtno è dispari
                beqz $t5,salto      #Salto se non è dispari
                sw $t4,vOUT($t3)    #archiviazione del valore nel vettore di ausilio
                add $t2,$t2,1       #incremwnto indice vOUT
salto:
                add $t0,$t0,1       #incremento indice vIN
                bne $t0,$t9,ciclo   #confronto fine vettore

                li $v0,10            #stampa terminazione
                syscall

                .data
                vIN: .word 3,5,8,10,1  #vettore da analizzare
                vOUT: .space 20        #vettore di ausilio
                numelem:.word 5        #Numero di elementi dei vettore
```


Stringa

Stringa

Generalità

Una stringa è un vettore di valori numerici,di dimensione di un byte, che hanno il significato di carattere alfanumerico

- Le stringhe sono sequenze numeriche (vettore) di caratteri con un terminatore **NULL** (il valore 0)
- La stringa priva di caratteri, la **stringa vuota**,è quella formata dal solo terminatore NULL (il valore 0)

Indice	*Posizione*	Elemento in memoria	Rappresentazione ASCII
0	*100*	**67**	**C**
1	*101*	**73**	**I**
2	*102*	**65**	**A**
3	*103*	**79**	**O**
4	*104*	**0**	**NULL**

Stringa

Definizione

È possibile definire una stringa con il terminatore NULL sfruttando la direttiva .**ASCIIZ** o senza terminatore NULL sfruttando la direttiva **.ASCII**

Stringa:.asciiz “CIAO”

Posizione	Elemento in memoria	Rappresentazione ASCII
100	**67**	**C**
101	**73**	**I**
102	**65**	**A**
103	**79**	**O**
104	**0**	**NULL**

Stringa:.ascii “CIAO”

Posizione	Elemento in memoria	Rappresentazione ASCII
100	**67**	**C**
101	**73**	**I**
102	**65**	**A**
103	**79**	**O**
104	***Qualsiasi valore***	***Qualsiasi carattere***

Stringa

Codifica ASCII

Dec	Sym	Dec	Char	Dec	Char	Dec	Char
0	NUL	32		64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	TAB	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	□

Dec	Char	Dec	Char	Dec	Char	Dec	Char
128	Ç	160	á	192	+	224	Ó
129	ü	161	í	193	-	225	ß
130	é	162	ó	194	-	226	Ô
131	â	163	ú	195	+	227	Ò
132	ä	164	ñ	196	-	228	õ
133	à	165	Ñ	197	+	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	+	232	Þ
137	ë	169	®	201	+	233	Ú
138	è	170	¬	202	-	234	Û
139	ï	171	½	203	-	235	Ù
140	î	172	¼	204	¦	236	ý
141	ì	173	¡	205	-	237	Ý
142	Ä	174	«	206	+	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	_	208	¶	240	Û
145	æ	177	_	209	Ð	241	±
146	Æ	178	_	210	Ê	242	_
147	ô	179	¦	211	Ë	243	¾
148	ö	180	¦	212	È	244	¶
149	ò	181	Á	213	i	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	¦	217	+	249	¨
154	Ü	186	¦	218	+	250	·
155	ø	187	+	219	_	251	¹
156	£	188	+	220	_	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	_
159	ƒ	191	+	223	_	255	

Stringa

Esercizio

Definita una stringa in memoria calcolare il numero di occorrenze di A

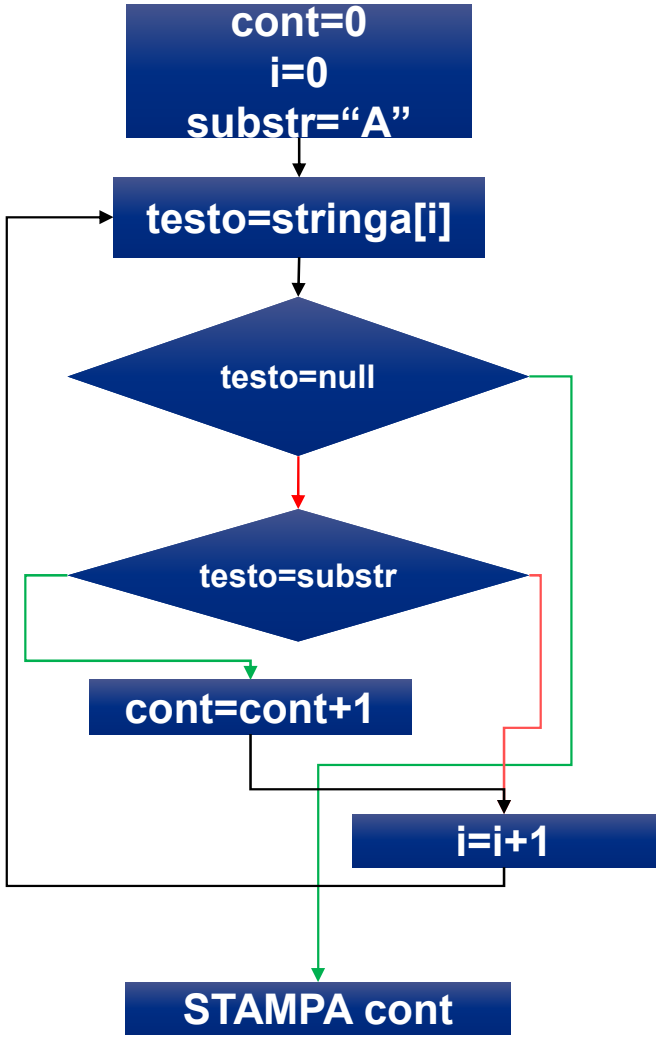
Esempio:

ALESSANDRO BERGONZONI POLIARTISTA

4

Stringa

Esercizio (numero di occorrenze di una lettera)




```
                .text
                .globl main
main:
                #CONTEGGIO DI "A" IN UNA STRINGA DEFINITA IN MEMORIA
                la $t0,stringa          #conservo l'indirizzo iniziale della stringa
                li $t1,65               #Valore del carattere "A" in ASCII
                li $t7,0                #Azzeramento registro dei risultati
ciclo:
                lb $t2,($t0)            #Prelevo il carattere
                beqz $t2,fine           #se è finita la stringa esce dal ciclo
                beq $t2,$t1,trovato     # salta se carattere trovato è A (non aggiorna il contatore di A)
                add $t0,$t0,1           # incremento dell'indice
                j ciclo
trovato:
                add $t7,$t7,1           #incrementa e aggiorna il contatore delle A presenti
                add $t0,$t0,1           #incrementa l'elemento stringa
                j ciclo
fine:
                move $a0,$t7            #stampa delle occorrenze
                li $v0,1
                syscall
                li $v0,10
                syscall
                .data
                stringa:.asciiz "BUONA GIORNATA A TUTTI GLI STUDENTI"
```

Stringa

Esercizio(palindromicità stringa)

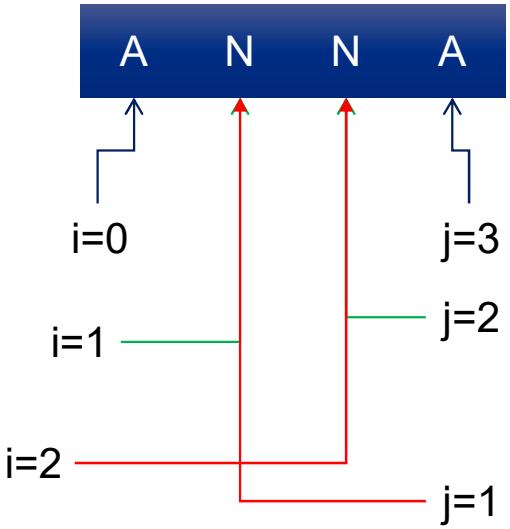
Stringa palindroma

Esempio:

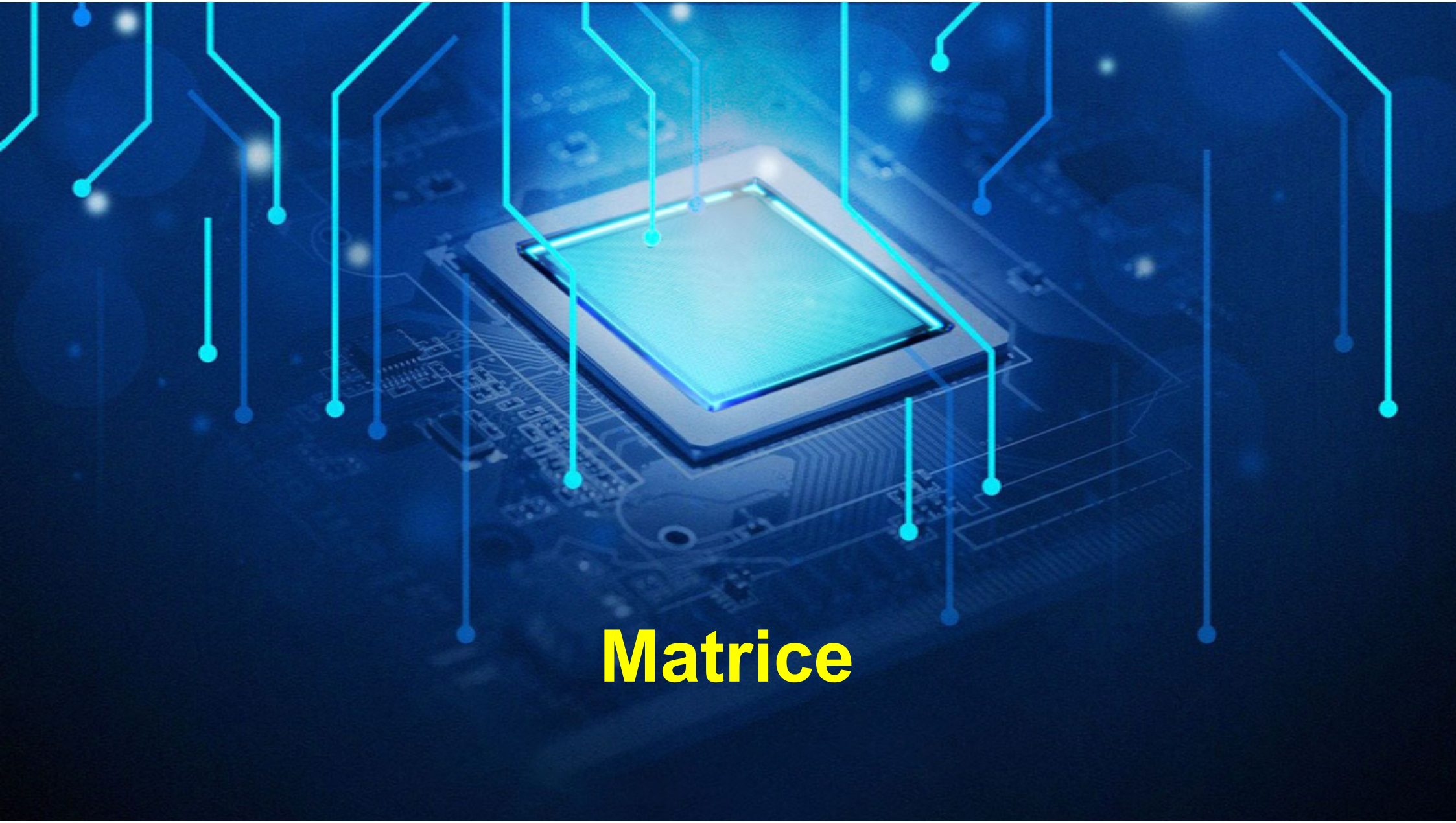
ANNA	OTTA
Palindorma	Non palindroma

Stringa

Esercizio (palindromicità)




```
                .text
                .globl main
main:
                la $a0,stringa          #conserva l'indirizzo iniziale della stringa
lung:
                lb $t0,($a0)            #conta il numeri di caratteri della stringa
                add $a0,$a0,1
                bnez $t0,lung
                sub $a0,$a0,2           #Mi posizione all'ultimo carattere
                move $a1, $a0           #copio indirizzo del'ultimo carattere [indice J]
                la $a0,stringa          #conservo l'indirizzo iniziale della stringa [indice I]
ciclo:
                lb $t0,($a0)
                lb $t1,($a1)
                bne $t0,$t1,nonpal      #confronto v[I]==v[J] Se diversi NON PALINDROMA
                add $a0,$a0,1                   #I=I+1
                sub $a1,$a1,1                   #J=J-1
                blt $a1,$a0,pal                 #Se J<I finisco quindi STRINGA PALINDROMA
                j ciclo
nonpal:
                la $a0,nonpalindroma#STAMPA MESSAGGIO "NON PALINDROMA E FINE"
                li $v0,4
                syscall
                j fine
pal:
                la $a0,palindroma       #STAMPA MESSAGGIO "PALINDROMA E FINE"
                li $v0,4
                syscall
                                                .data
fine:                                           stringa: .asciiz "ANNA"
                li $v0,10                       palindroma: .asciiz "STRINGA PALINDROMA"
                syscall                         nonpalindroma: .asciiz  "STRINGA NON PALIDROMA"
```


Matrice

Matrice

Generalità

Una matrice è una tabella di dati omogenei *(dello stesso tipo)* costituita da *r* righe ed *c* colonne M_{rxc}

- Gli *elementi* di una matrice hanno tutti la stessa dimensione (*esize*).
- La posizione (*indice riga/ indice colonna*) identifica i singoli elementi

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}
a_{51}	a_{52}	a_{53}	a_{54}

1	32	81	12
123	490	72	345
100	58	63	44
9	34	556	33
8	56	77	22

Matrice

Definizione

Il linguaggio assemblativo non consente una particolare definizione di matrice: si utilizza un'array con lunghezza *m*x*n* e un indice per accedere alla *i*-esima posizione

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}
a_{51}	a_{52}	a_{53}	a_{54}

a_{11}
a_{12}
a_{13}
a_{14}
a_{21}
a_{22}
a_{23}
a_{24}
a_{31}
a_{32}
a_{33}
a_{34}
a_{41}
a_{42}
a_{43}
a_{44}
a_{51}
a_{52}
a_{53}
a_{54}

```
.text
li $t0,0          # in $t0 viene posto l'indice di riga i che assume valori da 0 a 4
li $t1,0          # in $t1 viene posto l'indice di colonna j che assume valori da 0 a 3
lw $t2,nrig       # in $t0 viene posto il numero di righe della matrice
lw $t3,ncol       # in $t1 viene posto il numero di colonne della matrice
…

.data
nrig: .word 5     # definizione del valore del numero di righe
ncol: .word 4     # definizione del valore del numero di colonne
matrice: .word 0:20 #area dell'area di memoria che conterrà la matrice 20 words inizializzate a zero
```

Matrice

Indice

Gli indici relativi della matrice iniziano dal valore 0 che corrisponde all'elemento (0,0)

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}
a_{40}	a_{41}	a_{42}	a_{43}

Gli indici relativi poi devono essere moltiplicati per la dimensione dell'elemento
(4 per word, 2 per halfword, 1 per byte)

Matrice

Accesso elemento

L'accesso ad un elemento (r,c) di una matrice di R righe e C colonne richiede il seguente calcolo:

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}
a_{51}	a_{52}	a_{53}	a_{54}

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

Posizione elemento :$a_{r,c}$= C(r-1)+(c-1)

ESEMPIO:

a_{43}=4(3)+2=14

Matrice

Gestione matrice (scansione)

```
for (i=1;i<R;i++)
        {
        for (j=1;j<C;j++)
                {
                elemento= M[i,j]
                elaborazione dato
                }
        print ("\n");
        }
```

```
main:
        li $t0,1 #indice r
        li $t1,1 #indice c
        lw $t2,R #numero righe R
        lw $t3,C #numero colonne C
analisi_riga:
        li $t1,1
analisi_colonna:
        sub $t6,$t0,1           #calcolo elemento r,c
        mul $t9,$t6,$t3         #
        sub $t7,$t1,1           #r,c= C(r-1)+(c-1)
        add $t9,$t9,$t7         #
        lb $t8,matrice($t9)     #prelievo elemento
                                #elaborazione
        addi $t1,$t1,1          #incremento colonna
        ble $t1,$t3, analisi_colonna
        la $a0,riga             #stampa andata a capo (nuova riga)
        li $v0,4
        syscall
        addi $t0,$t0,1          #incremento riga
        ble $t0,$t2, analisi_riga
```

Matrice

- Esempio: Stampa degli elementi di un matrice nella rappr. canonica

```
for (i=1;i<R;i++)
        {
        for (j=1;j<C;j++)
                {
                elemento= M[i,j];
                print(elemento);
                print("\t");
                }
        print ("\n");
        }
```

```
                .text
                .global main
main:
                li $t0,1 #indice r
                li $t1,1 #indice c
                lw $t2,R #numero righe R
                lw $t3,C #numero colonne C
analisi_riga:
                li $t1,1
analisi_colonna:
                sub $t6,$t0,1                   #calcolo elemento r,c
                mul $t9,$t6,$t3                 #
                sub $t7,$t1,1                   #               r,c= C(r-1)+(c-1)
                add $t9,$t9,$t7                 #
                mul $t9,$t9,2                   #moltiplicazione dimensione (2=halfword)
                lh $t8,matrice($t9)             #prelievo elemento
                move $a0,$t8                    #stampa elemento
                li $v0,1                        #
                syscall                         #
                la $a0,tabulato                 #stampa tabulato
                li $v0,4
                syscall
                addi $t1,$t1,1                  #incremento colonna
                ble $t1,$t3, analisi_colonna
                la $a0,riga             #stampa andata a capo (nuova riga)
                li $v0,4
                syscall
                addi $t0,$t0,1                  #incremento riga
                ble $t0,$t2, analisi_riga
                li $v0,10
                syscall
```

```
.data
matrice: .half 2:20
R: .word 4
C: .word 5
tabulato:.asciiz "\t"
riga:.asciiz "\n"
```

Matrice

Esempio: stampa della somma degli elementi lungo le diagonali di una matrice quadrata

Data una matrice 4x4 di interi (word) riportare la somma degli elementi presenti lungo le diagonali

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

ESEMPIO:

45	2534643	34453	3
56474	6	337	8232
943634	670	104	23415
60000	67324	432435	10000

(45+6+104+10000)+(3+337+670+60000)
10155+61010=**71165**

Matrice

Esempio: stampa della somma degli elementi lungo le diagonali di una matrice quadrata

```
        .text
        .global main
main:
        xor $s0,$s0,$s0         #contatore

        li $t0,1                #indice r
        li $t1,1                #indice c
        lw $t2,R                #numero righe R
        lw $t3,C                #numero colonne C
ciclo1:             #analisi diagonale principale (sin->dex)
        sub $t6,$t0,1           #calcolo elemento r,c
        mul $t9,$t6,$t3         #
        sub $t7,$t1,1           #           r,c= C(r-1)+(c-1)
        add $t9,$t9,$t7         #
        mul $t9,$t9,4           # dimensione
        lw $t8,matrice($t9)     #prelievo elemento
        add $s0,$s0,$t8         #somma al contatore
        add $t0,$t0,1           #incremento riga
        add $t1,$t1,1           #incremento colonna
        ble $t1,$t3,ciclo1      #aconfronto con fine colonna (i,j)=(R,C
        li $t0,1 #indice r
        li $t1,4 #indice c
        lw $t2,R #numero righe R
        lw $t3,C #numero colonne C

ciclo2:                         #analisi diagonale secondaria
        sub $t6,$t0,1                   #calcolo elemento r,c
        mul $t9,$t6,$t3         #
        sub $t7,$t1,1                   #           r,c= C(r-1)+(c-1)
        add $t9,$t9,$t7         #
        mul $t9,$t9,4                   # dimensione
        lw $t8,matrice($t9)     #prelievo elemento
        add $s0,$s0,$t8         #somma al contatore
        add $t0,$t0,1                   #incremneto riga
        sub $t1,$t1,1                   #decremento colonna
        ble $t0,$t2,ciclo2      #confornto con fine (i,j)=(R,1)

        move $a0,$s0
        li $v0,1
        syscall
        li $v0,10
        syscall
```

```
.data
matrice: .word 45,2534643,34453,3,56474,6,337,8232,943634,670,104,23415,60000,67324,432435,10000
R: .word 4
C: .word 4
```

FINE