

Calcolatori Elettronici

Esercitazione 9

M. Sonza Reorda – M. Monetti

M. Rebaudengo – R. Ferrero

L. Sterpone – E. Vacca

Politecnico di Torino

Dipartimento di Automatica e Informatica

Tutoring finale

- Disponibili 2 sessioni di tutoring da remoto
 - Martedì **03/06** h.17-19
 - Mercoledì **04/06** h.17-19

Esercizio 1

- Sono date due matrici quadrate contenenti numeri con segno, memorizzate per righe, di DIMxDIMelementi.
- Si scriva una procedura **variazione** in grado di calcolare la variazione percentuale (troncata all'intero) tra gli elementi di indice corrispondente della riga *i* della prima matrice ($[i, 0]$, $[i, 1]$, $[i, 2]$...) e della colonna *i* della seconda ($[0, i]$, $[1, i]$, $[2, i]$...).
- La variazione percentuale è calcolata come segue:
$$\text{variazione} = (\text{valore2} - \text{valore1}) \cdot 100 / \text{valore1}$$

Esercizio 1 [cont.]

- La procedura riceve i parametri nel seguente ordine:
 - 1. indirizzo della prima matrice
 - 2. indirizzo della seconda matrice
 - 3. indirizzo del vettore risultato
 - 4. dimensione DIM
 - 5. indice i
- Esempio: date due matrici 3x3 e con $i = 2$

$$\begin{bmatrix} 4 & -45 & 15565 \\ 6458 & 4531 & 124 \\ -548 & 2124 & 31000 \end{bmatrix}$$

$$\begin{bmatrix} 6 & -5421 & -547 \\ -99 & 4531 & 1456 \\ 4592 & 118 & 31999 \end{bmatrix}$$

il risultato è 0, -31, 3

Esercizio 1 [cont.]

- Di seguito un esempio di programma chiamante:

```
.equ DIM, 3

.data
mat1:.word 4, -45, 15565, 6458, 4531, 124, -548, 2124, 31000
mat2:.word 6, -5421, -547, -99, 4531, 1456, 4592, 118, 31999
indice:.word 2
vet_out: .zero DIM*4

.text
main:
    la a0, mat1
    la a1, mat2
    la a2, vet_out
    li a3, DIM
    la t0, indice
    lw a4, 0(t0)
    jal variazione

    li a7, 10
    ecall
```

[1]-Soluzione [cont.]

```
.equ DIM, 3
.data
mat1:      .word 4, -45, 15565,      6458, 4531, 124,      -548, 2124, 31000
mat2:      .word 6, -5421, -547,      -99, 4531, 1456,      4592, 118, 31999
indice:    .word 2
vet_out:   .zero DIM*4

.text
main:      la a0, mat1
           la a1, mat2
           la a2, vet_out
           li a3, DIM
           la t0, indice
           lw a4, 0(t0)
           jal variazione
           li a7, 10
           ecall
```

[1]-Soluzione [cont.]

variazione:

```
addi sp, sp, -12
sw ra, 0(sp)           # salvo ra perché la procedura non è leaf
sw s0, 4(sp)
sw s1, 8(sp)
```

```
slli s0, a3, 2
mul t1, a4, s0
add a0, a0, t1         # indirizzo riga i della matrice 1
```

```
slli a4, a4, 2
add a1, a1, a4         # indirizzo colonna i della matrice 2
```

```
li s1, 0               # contatore
```

```
ciclo1:  addi sp, sp, -16
          sw a0, 0(sp)
          sw a1, 4(sp)
          sw a2, 8(sp)           # salvo i registri a2 e a3 nello stack
          sw a3, 12(sp)         # perché calcoloVariazione potrebbe modificarli
          lw a0, 0(a0)          # primo argomento: valore1
          lw a1, 0(a1)          # secondo argomento: valore2
          jal calcoloVariazione
```

[1]-Soluzione

```
lw a1, 4(sp)
lw a2, 8(sp)
lw a3, 12(sp)
sw a0, 0(a2)
lw a0, 0(sp)
addi sp, sp, 16
```

```
# il valore di ritorno di calcoloVariazione salvato nel vettore a2
# ripristina a0
```

```
addi a0, a0, 4
add a1, a1, s0
addi a2, a2, 4
addi s1, s1, 1
bne s1, a3, ciclo1
```

```
lw ra, 0(sp)
lw s0, 4(sp)
lw s1, 8(sp)
addi sp, sp, 12
jr ra
```

calcoloVariazione:

```
# lavoro nell'ipotesi di non avere overflow
```

```
sub t0, a1, a0
li t1, 100
mul t0, t0, t1
div a0, t0, a0
jr ra
```


Esercizio 2

- Si scriva una procedura **sostituisci** in grado di espandere una stringa precedentemente inizializzata sostituendo tutte le occorrenze del **carattere %** con un'altra stringa data.
- Siano date le seguenti tre stringhe in memoria:
 - ***str_orig***, corrispondente al testo da espandere
 - ***str_sost***, contenente il testo da sostituire in ***str_orig*** al posto di %
 - ***str_new***, che conterrà la stringa espansa (si supponga che abbia dimensione sufficiente a contenerla).

Esercizio 2 [cont.]

La procedura riceve gli indirizzi delle 3 stringhe (nell'ordine indicato), e restituisce la lunghezza della stringa finale.

Di seguito un esempio di funzionamento:

- **Stringa originale:** "% nella citta'dolente, % nell'eterno dolore, % tra la perduta gente"
- **Stringa da sostituire:** "per me si va"
- **Risultato:** "per me si va nella citta'dolente, per me si va nell'eterno dolore, per me si va tra la perduta gente"

Esercizio 2 [cont.]

Esempio programma **Main**

```
.data
str_orig: .string"% nella citta'dolente, % nell'eterno dolore, % tra la
perduta gente"
str_sost: .string"per me si va"
str_new: .zero 200

.text
main:
    la a0, str_orig
    la a1, str_sost
    la a2, str_new
    jal sostituisce
    li a7, 10
    ecall
```

[2]-Soluzione [cont.]

```
.equ '%', 37
.data
str_orig: .string "% nella citta' dolente, % nell'eterno dolore, % tra la perduta gente"
str_sost: .string "per me si va"
str_new:  .zero 200

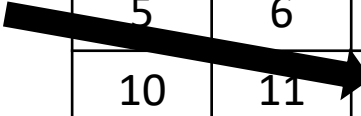
.text
main:    la a0, str_orig
        la a1, str_sost
        la a2, str_new
        jal sostituisce
        la a0, str_new          # stampa la stringa finale
        li a7, 4
        ecall
        li a7, 10
        ecall
```

[2]-Soluzione

```
sostituisci: addi sp, sp, -4
             sw a2, 0(sp)                # salvataggio indirizzo str_new (per calcolo
lunghezza)
             li t4, '%'
ciclo1:      lbu t0, 0(a0)
             beq t0, zero, fine          # controllo fine stringa
             bne t0, t4, copia           # controllo carattere da sostituire
             mv t1, a1                   # sostituzione
ciclo2:      lbu t2, 0(t1)
             beq t2, zero, next
             sb t2, 0(a2)
             addi t1, t1, 1
             addi a2, a2, 1
             j ciclo2
copia:       sb t0, 0(a2)                # copia caratteri stringa
             addi a2, a2, 1
next:        addi a0, a0, 1
             j ciclo1
fine:        sb zero, 0(a2)
             lw t0, 0(sp)                # calcolo lunghezza della nuova stringa
             addi sp, sp, 4
             sub a0, a2, t0
             jr ra
```

Esercizio 3

- Sia data una matrice di byte, contenente numeri senza segno.
- Si scriva una procedura **contaVicini** in grado di calcolare (e restituire come valore di ritorno) la somma dei valori contenuti nelle celle adiacenti ad una determinata cella.
- La procedura **contaVicini** riceve i seguenti parametri:
 - indirizzo della matrice
 - numero progressivo della cella X, così come indicato nell'esempio a fianco
 - numero di righe della matrice
 - numero di colonne della matrice.



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

Esercizio 3 [cont.]

- Di seguito un esempio di programma chiamante:

```
.equ RIGHE, 4
```

```
.equ COLONNE, 5
```

```
.data
```

```
matrice: .byte 0, 1, 3, 6, 2, 7, 13, 20, 12, 21, 11, 22, 10, 23, 9, 24, 8, 25, 43, 62
```

```
.text
```

```
main:la a0, matrice
```

```
li a1, 12
```

```
li a2, RIGHE
```


```
li a3, COLONNE
```

```
jalcontaVicini
```

```
li a7, 10
```

```
ecall
```

0	1	3	6	2
7	13	20	12	21
11	22	10	23	9
24	8	25	43	62



il valore restituito è 166, pari a
 $13 + 20 + 12 + 22 + 23 + 8 + 25 + 43$

[3]-Soluzione [cont.]

```
.equ RIGHE, 4
.equ COLONNE, 5
.data
matrice: .byte 0, 1, 3, 6, 2, 7, 13, 20, 12, 21, 11, 22, 10, 23, 9, 24, 8, 25, 43, 62

.text
main:    la a0, matrice
         li a1, 12
         li a2, RIGHE
         li a3, COLONNE
         jal contaVicini
         li a7, 1           # stampa il valore di ritorno
         ecall
         li a7, 10
         ecall

contaVicini:
         addi sp, sp, -4
         sw s0, 0(sp)
         li s0, 0           # somma delle celle vicine

         divu t0, a1, a3     # t0: indice riga
         remu t1, a1, a3     # t1: indice colonna
         li t6, -1
```


[3]-Soluzione [cont.]

```
# indice riga sopra
    addi t2, t0, -1
    bne t2, t6, indiceRigaSotto
    mv t2, zero

indiceRigaSotto:
    addi t3, t0, 1
    bne t3, a2, indiceColonnaASinistra
    addi t3, a2, -1

indiceColonnaASinistra:
    addi t4, t1, -1
    bne t4, t6, indiceColonnaADestra
    mv t4, zero

indiceColonnaADestra:
    addi t5, t1, 1
    bne t5, a3, indiciCelle
    addi t5, a3, -1
```

[3]-Soluzione [cont.]

indiciCelle:

```
mul t1, t2, a3
add t0, t1, t4      # indice dell'elemento a sinistra nella riga sopra
add t1, t1, t5      # indice dell'elemento a destra nella riga sopra

mul t2, t3, a3
add t2, t2, t4      # indice dell'elemento a sinistra nella riga sotto

add t0, t0, a0      # somma l'indirizzo iniziale della matrice
add t1, t1, a0
add t2, t2, a0
add a1, a1, a0
```

cicloEsterno:

```
mv t3, t0
```

cicloInterno:

```
beq t3, a1, saltaElemento
lb t4, 0(t3)
add s0, s0, t4
```

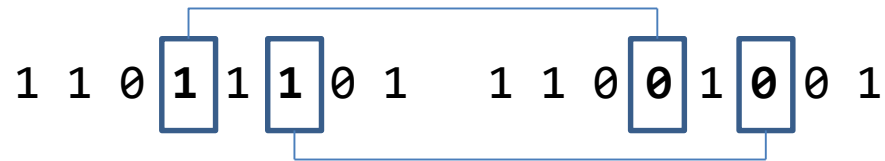
[3]-Soluzione

```
addi t3, t3, 1
bleu t3, t1, cicloInterno
add t0, t0, a3
add t1, t1, a3
bleu t0, t2, cicloEsterno
```

```
mv a0, s0
lw s0, 0(sp)
addi sp, sp, 4
jr ra
```

Esercizio 4

La distanza di Hamming tra due stringhe di ugual lunghezza è pari al numero di posizioni nelle quali i simboli corrispondenti sono diversi. In altri termini, la distanza di Hamming misura il numero di sostituzioni necessarie per convertire una stringa nell'altra, o il numero di modifiche necessarie per trasformare una stringa nell'altra. Ad esempio, si consideri la distanza di Hamming binaria tra i seguenti due interi:



Il risultato in questo caso è 2.

Si scriva una procedura `CalcolaDistanzaH` in linguaggio che calcoli la distanza di Hamming binaria tra gli elementi di indice corrispondente di due vettori di *word* di lunghezza `DIM` (dichiarato come costante).

Esempio (valori in decimale e binario):

vet1	vet2	risultato
56 (0000 0000 0011 1000)	1 (0000 0000 0000 0001)	4
12 (0000 0000 0000 1100)	0 (0000 0000 0000 0000)	2
98 (0000 0000 0110 0010)	245 (0000 0000 1111 0101)	5
129 (0000 0000 1000 0001)	129 (0000 0000 1000 0001)	0
58 (0000 0000 0011 1010)	12 (0000 0000 0000 1100)	4

[4]-Soluzione [cont.]

```
.equ 'DIM',5
```

```
.data
```

```
vet1:      .word      56, 12, 98, 129, 58
```

```
vet2:      .word      1, 0, 245, 129, 12
```

```
risultato: .zero      20
```

```
.text
```

```
main:
```

```
la    a0,vet1
```

```
la    a1,vet2
```

```
la    a2,risultato
```

```
li    a3,'DIM'
```

```
jal   calcola_distanzaH
```

```
li a7,10
```

```
ecall
```

[4]-Soluzione [cont.]

calcola_distanzaH:

a0=vet1 a1=vet2 a2=risultato a3=DIM

Ciclo

li s0,0 # t0 contatore Cicli

ciclo:

beq s0,a3,fine_ciclo

calcoloH:

lw s7,0(a0)

lw s8,0(a1)

xor s2,s7,s8

andi s3,s3,0 # azzeramento risultato

andi s4,s4,0 # azzeramento indice

li s5,1 # mask per lettura bit a 1

cicloH: and s6,s2,s5

beqz s6,nextH

addi s3,s3,1

nextH: li s11,1

sll s5,s5,s11

addi s4,s4,1

li s11,16

bne s4,s11, cicloH

[4]-Soluzione

```
# In s3 il risultato  
sw    s3,0(a2)
```

```
addi   s0,s0,1
```

```
addi   a0,a0,4
```

```
addi   a1,a1,4
```

```
addi   a2,a2,4
```

```
j ciclo
```

```
fine_ciclo:
```

```
jr ra
```