

# Calcolatori Elettronici

## Esercitazione 9

M. Sonza Reorda – M. Monetti

M. Rebaudengo – R. Ferrero

L. Sterpone – M. Grosso

Politecnico di Torino

Dipartimento di Automatica e Informatica

# Esercizio 1

Sono date due matrici quadrate contenenti numeri con segno, memorizzate per righe, di DIMxDIM elementi. Si scriva una procedura **Variazione** in linguaggio MIPS in grado di calcolare la variazione percentuale (troncata all'intero) tra gli elementi di indice corrispondente della *riga*  $l$  della prima matrice ( $[l, 0]$ ,  $[l, 1]$ ,  $[l, 2]$ ...) e della *colonna*  $l$  della seconda ( $[0, l]$ ,  $[1, l]$ ,  $[2, l]$ ...). Ad esempio, nel caso di due matrici 3x3 e con  $l = 2$ :

$$\begin{bmatrix} 4 & -45 & 15565 \\ 6458 & 4531 & 124 \\ -548 & 2124 & 31000 \end{bmatrix} \quad \begin{bmatrix} 6 & -5421 & -547 \\ -99 & 4531 & 1456 \\ 4592 & 118 & 31999 \end{bmatrix}$$

il risultato è 0, -31, 3

# Esercizio 1: implementazione

- La variazione percentuale è calcolata come segue:

$$Variazione = (Val2 - Val1) \cdot 100 / Val1$$

- La procedura riceve i seguenti parametri:
  - L'indirizzo della prima matrice mediante \$a0
  - L'indirizzo della seconda matrice mediante \$a1
  - L'indirizzo del vettore risultato mediante \$a2
  - La dimensione DIM tramite \$a3
  - L'indice i per mezzo dello stack.

# 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 quindi le seguenti tre stringhe in memoria:
  - `str_orig`, corrispondente al testo compresso da espandere
  - `str_sost`, contenente la 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).
- 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.]

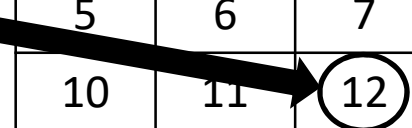
- La procedura riceve gli indirizzi delle 3 stringhe attraverso i registri \$a0, \$a1 e \$a2, e restituisce la lunghezza della stringa finale attraverso \$v0.
- Le stringhe sono terminate dal valore ASCII 0x00.
- Di seguito un esempio di programma chiamante:

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

                .text
                .globl main
                .ent main
main:           [...]
                la $a0, str_orig
                la $a1, str_sost
                la $a2, str_new
                jal sostituisci
                [...]
```

# 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.
- La procedura deve essere conforme allo standard per quanto riguarda passaggio di parametri, valore di ritorno e registri da preservare.



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:

RIGHE = 4

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

.globl main

.ent main

main: [...]

la \$a0, matrice

li \$a1, 12

li \$a2, RIGHE

li \$a3, COLONNE

jal contaVicini

[...]

.end main

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  
 $13 + 20 + 12 + 22 + 23 + 8 + 25 + 43$

# Esercizio 4

- Il gioco della vita sviluppato dal matematico John Conway si svolge su una matrice bidimensionale.
- Le celle della matrice possono essere vive o morte.
- I vicini di una cella sono le celle ad essa adiacenti.
- La matrice evolve secondo le seguenti regole:
  - una cella con meno di due vicini vivi muore (isolamento)
  - una cella con due o tre vicini vivi sopravvive alla generazione successiva
  - una cella con più di tre vicini vivi muore (sovrappopolazione)
  - una cella morta con tre vicini vivi diventa viva (riproduzione).
- L'evoluzione avviene contemporaneamente per tutte le celle.



# Esercizio 4 [cont.]

- Si scriva un programma in MIPS in grado di giocare al gioco della vita.
- Il programma principale esegue un ciclo di N iterazioni; ad ogni iterazione chiama la procedura **evoluzione** che determina il nuovo stato delle celle nella matrice.
- La procedura **evoluzione** riceve i seguenti parametri:
  - indirizzo di una matrice di byte, le cui celle hanno solo due valori: vivo (1) e morto (0)
  - indirizzo di una seconda matrice di byte non inizializzata di pari dimensioni
  - numero di righe delle due matrici
  - numero di colonne delle due matrici.

## Esercizio 4 [cont.]

- La procedura **evoluzione** effettua un ciclo su tutte le celle della prima matrice:
  - per ogni cella, chiama la procedura **contaVicini**, implementata nell'esercizio precedente, per contare il numero di vicini
  - in base allo stato della cella e al suo numero di vicini, setta lo stato futuro della corrispondente cella nella seconda matrice.
- Al termine del ciclo, la procedura **evoluzione** chiama la procedura **stampaMatrice** che visualizza a video la seconda matrice, passando i seguenti parametri:
  - indirizzo della matrice
  - numero di righe della matrice
  - numero di colonne della matrice.
- Tutte le procedure devono essere conformi allo standard.