Calcolatori Elettronici
Esercitazione 3

M. Sonza Reorda – M. Monetti
M. Rebaudengo – R. Ferrero
L. Sterpone – M. Grosso

Politecnico di Torino
Dipartimento di Automatica e Informatica

Obiettivi

- Input e output robusti
- Operazioni di moltiplicazione e divisione

Esercizio 1

- La system call 5 permette di leggere in input un numero intero.
- Cosa succede se l'utente introduce da tastiera un carattere non numerico?
- Si realizzi un programma per effettuare la verifica dell'input da tastiera per la lettura robusta di un numero intero positivo.
- Il programma deve leggere singoli caratteri tramite la system call 12, verificare se sono cifre e terminare quando è letto '\n'.

Soluzione

```
                .data
messageInput:   .asciiz "Inserisci un numero: "
messageError:   .asciiz "\nL'input non e' un numero valido."
messageOk:      .asciiz "L'input e' corretto."
messageEmpty:   .asciiz "Non e' stato inserito nessun numero!"

                .text
                .globl main
                .ent main
main:           la $a0, messageInput
                li $v0, 4
                syscall
                move $t0, $0    #contatore numero di caratteri letti
loop:           li $v0, 12      # lettura di un carattere
                syscall
```

Soluzione

```
                beq $v0, '\n', exitLoop
                blt $v0, '0', notANumber
                bgt $v0, '9', notANumber
                addi $t0, $t0, 1
                b loop
exitLoop:       beq $t0, 0, noInput
                la $a0, messageOk
                b printMessage
noInput:        la $a0, messageEmpty
                b printMessage
notANumber:     la $a0, messageError
printMessage:   li $v0, 4
                syscall
                li $v0, 10
                syscall
                .end main
```

Esercizio 2

- Si modifichi l'esercizio precedente per la lettura robusta di un intero positivo tramite la system call 12.
- Oltre a verificare se i caratteri introdotti siano cifre, il programma deve controllare se il numero sia rappresentabile su 4 byte.
- Il programma termina quando è letto '\n'; il numero introdotto in input deve essere stampato a video tramite la system call 1.

Implementazione

- Per convertire una sequenza di caratteri in un intero si utilizza un ciclo. Dopo aver inizialmente azzerato un registro accumulatore, ad ogni iterazione:
  1. l'ultimo carattere letto è convertito in intero sottraendo al suo codice ASCII il valore '0'
  2. il valore nell'accumulatore è moltiplicato per 10
  3. si somma il valore calcolato all'accumulatore.
- Si noti che le operazioni ai punti 2 e 3 possono dare un *overflow*. In questo caso il programma deve stampare un opportuno messaggio.

Esempio

- L'utente inserisce i caratteri '3', '4', '6', '\n'
- Prima iterazione:
  1. valore letto = '3' – '0' = 3
  2. accumulatore * 10 = 0 * 10 = 0
  3. valore corrente = 0 + 3 = 3
- Seconda iterazione:
  1. valore letto = '4' – '0' = 4
  2. accumulatore * 10 = 3 * 10 = 30
  3. valore corrente = 30 + 4 = 34
- Terza iterazione:
  1. valore letto = '6' – '0' = 6
  2. accumulatore * 10 = 34 * 10 = 340
  3. valore corrente = 340 + 6 = 346
- Quarta iterazione:
  1. valore letto = '\n'
  2. Il programma termina e stampa a video 346

Soluzione

```
                .data
messageInput:   .asciiz "Inserisci un numero: "
messageError:   .asciiz "\nL'input non e' un numero valido."
messageOk:      .asciiz "L'input e' corretto: "
messageEmpty:   .asciiz "Non e' stato inserito nessun numero!"
messageOutput:  .asciiz "\nIl numero e' troppo grande."
ZERO = 0-'0'
                .text
                .globl main
                .ent main
main:           la $a0, messageInput
                li $v0, 4
                syscall
                move $t0, $0    #contatore numero di caratteri letti
                move $t1, $0    #valore introdotto
                li $t3, 10      #costante
```

Soluzione

```
loop:   li $v0, 12      # lettura di un carattere
        syscall
        beq $v0, '\n', exitLoop
        blt $v0, '0', notANumber
        bgt $v0, '9', notANumber
        addi $t0, $t0, 1
        # conversione del valore
        multu $t1, $t3
        mfhi $t1
        bne $t1, $0, overflow
        mflo $t1
        addi $t2, $v0, ZERO
        addu $t2, $t1, $t2
        bltu $t2, $t1, overflow
        move $t1, $t2
        j loop
```

Soluzione

```
exitLoop:       beq $t0, 0, noInput
                la $a0, messageOk
                li $v0, 4
                syscall
                li $v0, 1
                move $a0, $t1
                syscall
                j endProgram
noInput:        la $a0, messageEmpty
                j printMessage
notANumber:     la $a0, messageError
                j printMessage
overflow:       la $a0, messageOutput
printMessage:   li $v0, 4
                syscall
endProgram:     li $v0, 10
                syscall
                .end main
```

Esercizio 3

- Siano date tre variabili di tipo *byte* in memoria, che rappresentino rispettivamente il numero di giorni, ore e minuti passati da un certo istante T_0.
- Si calcoli il numero totale di minuti passati da T_0, e tale valore sia salvato nella variabile di tipo *word* `risultato`.
- È possibile ottenere *overflow* durante i calcoli?

Soluzione

```
            .data
risultato:  .space    4
giorni:     .byte   231
ore:        .byte    16
minuti:     .byte    47

            .text
            .globl main
            .ent main
main:
            lbu $t0, giorni     # conversione da giorni a ore
            mul $t1, $t0, 24
            lbu $t0, ore
            addu $t1, $t1, $t0  # somma ore
            mul $t1, $t1, 60    # conversione in minuti
            lbu $t0, minuti
            addu $t1, $t1, $t0
            sw $t1, risultato
            li $v0, 10
            syscall
            .end main
```

N.B. Estendendo gli operandi (unsigned) su 32 bit, anche nel peggiore dei casi (255, 255, 255) non è possibile ottenere overflow né nelle somme né nelle moltiplicazioni.

Esercizio 4

- Si scriva un programma che acquisisca `DIM` valori *word* e quindi ne calcoli la media (intera) e la stampi a video.
  - `DIM` deve essere dichiarato come costante
  - Si lavori nell'ipotesi di non avere *overflow* nei calcoli
  - Si noti il tipo di arrotondamento effettuato sul risultato della divisione.

Soluzione

```
            .data
DIM = 3
message_in:  .asciiz   "Inserire numeri: "
message_out: .asciiz   "Media: "

            .text
            .globl main
            .ent main
main:
            and $t0, $0, $0    # azzeramento contatore
            and $t1, $0, $0    # azzeramento accumulatore

            la $a0, message_in
            li $v0, 4
            syscall
```

Soluzione [cont.]

```
ciclo:      li  $v0, 5          # Read integer
            syscall             # system call (risultato in $v0)

            add $t1, $t1, $v0   # incremento accumulatore
            addiu $t0, $t0, 1   # incremento contatore

            bne $t0, DIM, ciclo

            la $a0, message_out
            li $v0, 4
            syscall

            div $t1, $t1, $t0

            li    $v0, 1        # Print integer
            move  $a0, $t1      # valore da stampare
            syscall             # system call

            li $v0, 10
            syscall
            .end main
```