Calcolatori Elettronici
Esercitazione 7

M. Sonza Reorda – M. Monetti
M. Rebaudengo – R. Ferrero
L. Sterpone – E. Vacca

Politecnico di Torino
Dipartimento di Automatica e Informatica

Esercitazione 7 - Obiettivi

- Stack
- Salvataggio dei registri nelle procedure
  - registri *callee-saved*: `s0`-`s11`, `ra`, `sp`
  - registri *caller-saved*: `t0`-`t6`, `a0`-`a7`
- Procedure l*eaf* e non *leaf*
- Passaggio dei parametri tramite stack

Esercizio 1

- Le ecall 34, 35 e 36 visualizzano il valore in a0 rispettivamente in esadecimale, binario e decimale (senza segno).
- Si vuole realizzare un programma che visualizzi un numero in una base B compresa tra 2 e 36
- Per verificare la correttezza del programma si può usare un convertitore online, ad esempio https://netlab.fauser.edu/s/convbasejs/converti.html

Esercizio 1

1. Scomposizione del numero nelle sue cifre tramite divisioni successive per la base *B*, salvando i resti e ripetendo l'operazione sul quoziente sino a che questo è diverso da zero
2. Visualizzazione dei resti in ordine inverso a quello di generazione

- Note:
  - per visualizzare le cifre in ordine inverso si usa lo stack, in quanto è una struttura LIFO
  - ciascuna cifra deve essere convertita in caratteri ASCII (0-9 e A-Z) e poi stampata con la ecall 11

Esercizio 1 - esempio

- Si vuole rappresentare 141592653 in base 33
- 141592653 / 33 = 4290686 con resto $15_{10} = F_{33}$
- 4290686 / 33 = 130020 con resto $26_{10} = Q_{33}$
- 130020 / 33 = 3940 con resto $0_{10} = 0_{33}$
- 3940 / 33 = 119 con resto $13_{10} = D_{33}$
- 119 / 33 = 3 con resto $20_{10} = K_{33}$
- 20 / 33 = 0 con resto $3_{10} = 3_{33}$
- Risultato: $141592653_{10} = 3KD0QF_{33}$

Esercizio 2

- Si consideri una sequenza di numeri naturali in cui, scelto il primo numero della sequenza c_0, gli elementi successivi sono così ottenuti:

$$c_{i+1} = \begin{cases} \frac{c_i}{2} & se\ c_i \text{ è } pari \\ 3 * c_i + 1 & se\ c_i \text{ è } dispari \end{cases}$$

- Si scriva una procedura `calcolaSuccessivo` che riceva tramite `a0` un numero naturale e calcoli l'elemento successivo della sequenza. Tale numero è stampato a video e restituito attraverso `a0`.

Esercizio 3 [cont.]

- La congettura di Collatz afferma che, per qualunque valore iniziale c_0, la sequenza definita nell'esercizio precedente raggiunge sempre il valore 1 passando attraverso un numero finito di elementi.
- Esempio: se c_0= 19, la sequenza è: 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. La sequenza contiene 21 elementi.
- La congettura di Collatz non è mai stata dimostrata, però è stata verificata sperimentalmente per tutti i numeri naturali fino a $87 * 2^{60} \approx 10^{21}$.

Esercizio 3

- Si scriva una procedura `sequenzaDiCollatz` che riceva tramite `a0` un numero naturale e restituisca attraverso `a0` il numero di elementi necessari per arrivare a 1.
- La procedura è costituita da un ciclo che a ogni iterazione calcola l'elemento successivo della sequenza, richiamando la procedura `calcolaSuccessivo` implementata nell'esercizio precedente.

Esercizio 4

- Si scriva una procedura `determinante2x2` che calcoli il valore del determinante di una matrice quadrata 2x2, ricevendo i 4 elementi tramite i registri **`a0`**, **`a1`**, **`a2`** e **`a3`** (matrice memorizzata per righe) e salvi il risultato in **`0`**

$$det = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} = a_1 b_2 - a_2 b_1$$

- Per validare la procedura, si scriva anche un programma chiamante che legga 4 valori salvati in memoria e lanci la procedura.
- Si assuma di non avere *overflow* nei calcoli.

Esercizio 5 [cont.]

- Si scriva una procedura `determinante3x3` in grado di calcolare il determinante di una matrice quadrata 3x3.

$$det = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \qquad \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \quad \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \quad \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

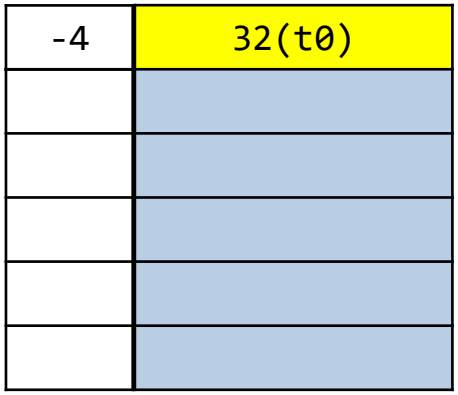
$$det = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} = a_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix} - b_1 \begin{vmatrix} a_2 & c_2 \\ a_3 & c_3 \end{vmatrix} + c_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}$$

Esercizio 5

- La procedura determinante3x3 riceve in input i 9 elementi della matrice. I primi 8 elementi sono passati attraverso i registri **a0-a7**, l'ultimo attraverso lo stack.
- La procedura determinante3x3 chiama 3 volte la procedura determinante2x2 implementata nell'esercizio 4.
- Per validare la procedura, si scriva anche un programma chiamante che legga 9 valori salvati in memoria e lanci la procedura.
- Si assuma di non avere overflow nei calcoli.

Soluzione [cont.]

```
                    .data
matrice:        .word 1, 41, 42, 13, 56, 23, 73, 9, 50
msg_output:     .string "Valore determinante: "
                .text
main:           la t0, matrice
                lw a0, 0(t0)     # A1
                lw a1, 4(t0)     # B1
                lw a2, 8(t0)     # C1
                lw a3, 12(t0)    # A2
                lw a4, 16(t0)    # B2
                lw a5, 20(t0)    # C2
                lw a6, 24(t0)    # A3
                lw a7, 28(t0)    # B3
                lw t1, 32(t0)    # C3  (SP)
                addi sp, sp, -4
                sw t1, 0(sp)
                jal determinante3x3
                addi sp, sp, 4
                mv t0, a0
                la a0, msg_output
                li a7, 4
                ecall
                mv a0, t0                   # intero da stampare
                li a7, 1
                ecall
                li a7, 10
                ecall
```



Soluzione [cont.]

```
determinante3x3:
                addi sp, sp, -20            # salva ra e s0-s3
                sw s0, 0(sp)
                sw s1, 4(sp)
                sw s2, 8(sp)
                sw s3, 12(sp)
                sw ra, 16(sp)
                mv s0, a0
                mv s1, a1
                mv s2, a2
                mv s3, a3

                mv a0, a4        # B2
                mv a1, a5        # C2
                mv a2, a7        # B3
                lw a3, 20(sp)    # C3
                # (a0*a3 – a1*a2) = B2*C3 – C2*B3
                jal determinante2x2
                mul s0, s0, a0

                mv a0, s3        # A2
                mv a1, a5        # C2
                mv a2, a6        # A3
                lw a3, 20(sp)    # C3
                # (a0*a3 – a1*a2) = A2*C3 – C2*A3
                jal determinante2x2
                mul s1, s1, a0
```

-4	32(t0)
-8	ra
-12	s3
-16	s2
-20	s1
-24	s0

$$a_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix}$$

$$b_1 \begin{vmatrix} a_2 & c_2 \\ a_3 & c_3 \end{vmatrix}$$

Soluzione

```
                mv a0, s3       # A2
                mv a1, a4       # B2
                mv a2, a6       # A3
                mv a3, a7       # B3
                # (a0*a3 – a1*a2) = A2*B3 – B2*A3
                jal determinante2x2
                mul s2, s2, a0
                add a0, s0, s2
                sub a0, a0, s1

                lw s0, 0(sp)    # rispristina ra e s0-s3
                lw s1, 4(sp)
                lw s2, 8(sp)
                lw s3, 12(sp)
                lw ra, 16(sp)
                addi sp, sp, 20
                jr ra

determinante2x2:
                mul t0, a0, a3
                mul t1, a1, a2
                sub a0, t0, t1
                jr ra
```

$$\cdot c_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}$$