

Esercizi sulla programmazione

Esercizio “sommatoria”

Scrivere un programma che, definito un **intero X non negativo**, calcola la **somma da 0 a X** (senza utilizzo di formule).

ES. se $X=6$, $\text{somma}=21$

Esercizio “sommatoria”

```
X: INT 6;  
ZERO: INT 0;  
UNO: INT 1;  
  
LOAD R0 X;  
LOAD R1 ZERO;  
LOAD R2 ZERO;  
LOAD R3 UNO;  
ciclo: COMP R0 R1;  
      BREQ fine;  
      ADD R2 R0;  
      SUB R0 R3;  
      BRANCH ciclo;  
fine: STORE R2 X;  
STOP;
```

Esercizio “fattoriale”

Definito un intero n , calcolare $n!$

Si definisce **n fattoriale** e si indica con **$n!$** il prodotto dei primi n numeri interi positivi minori o eguali di quel numero. In formule,

$$n! := \prod_{k=1}^n k = 1 \cdot 2 \cdot 3 \cdot \dots \cdot (n-1) \cdot n$$

per *definizione* si chiede poi che **$0!=1$**

Esercizio "fattoriale"

N: INT 4;
FATT: INT 1;
UNO: INT 1;
ZERO: INT 0;

LOAD R0 N;
LOAD R1 FATT;
LOAD R2 UNO;
LOAD R3 ZERO;
COMP R0 R3;
BRLT errore;
BREQ valuno;

ciclo: MUL R1 R0;
SUB R0 R2;
COMP R0 R2;
BRGT ciclo;
STORE R1 FATT;
STOP;
errore: STOP;
valuno: STORE R2 FATT;
STOP;

Domande (1)

- Quali sono le novità principali dell'Assembler rispetto al linguaggio macchina? (codici mnemonici, identificatori)
- In un programma assembler, perchè si attaccano etichette a certe istruzioni?

Domande (2)

- come si chiama in Assembler l'istruzione che trasferisce una parola dalla RAM ad un registro della CPU? E quella che compie il trasferimento inverso?
- In Assembler a cosa servono gli identificatori o variabili? (per rappresentare parole di memoria)

Esercizi I parte

Architettura del calcolatore

■ L'architettura di Von-Neumann

- RAM
- CPU
- Memoria secondaria
- Dispositivi input/output

Tavole di verita'

- Tavole di verita': **and, or, not** e circuiti corrispondenti
- Dalla **formula logica** alla **tavola di verita'**
- Dalla **tavola di verita'** al **circuito** corrispondente

Tavole di verita' (esercizi)

- Dare la **tavola di verita'** della formula
 $(\text{not}(A) \rightarrow B) \text{ or } (A \rightarrow \text{not}(B))$
e disegnare il **circuito** corrispondente

A=0,B=0: 1

A=0,B=1: 1

A=1,B=0: 1

A=1,B=1: 1

Rappresentazione dell'informazione - 1

- Rappresentazione binaria dei numeri decimali
 - Es.: **trovare il numero binario** corrispondente al numero decimale **20**
 - Sol.: **10100**
- Rappresentazione decimale dei numeri binari
 - Es.: **trovare il numero decimale** corrispondente al numero binario **1011**
 - sol.: **11**

Rappresentazione dell'informazione - 2

■ Somma binaria

- $10101 + 1011 =$

 $100000 \rightarrow \text{n.decimale } 32$

Rappresentazione dell'informazione - 3

■ Reali: da binario a decimale

- Es.: $11,11 \rightarrow 3,75$

■ Reali: da decimale a binario

- Metodo della divisione per la parte intera
- Metodo della moltiplicazione per la parte decimale
- Es.: $15,7 \rightarrow 1111, 1011$

Rappresentazione dell'informazione - 4

■ Da complemento a 2 a base 10

- $00010 \rightarrow \text{n.pos} \rightarrow 2$
- $11101 \rightarrow \text{n.neg} \rightarrow 00011 \rightarrow 3 \rightarrow -3$

■ Dati n bit

- N. **piu' grande** rappresentabile : $2^{(n-1)} - 1$
- N. **piu' piccolo** rappresentabile: $-2^{(n-1)}$
- Es. Con 4 bit posso rappresentare gli interi contenuti nell'intervallo $[-8, 7]$

Rappresentazione dell'informazione - 4

■ Da eccesso 8 a decimale

- $1111 \rightarrow 15 - 8 \rightarrow 7$
- $0110 \rightarrow 6 - 8 \rightarrow -2$

■ Da decimale a eccesso 8

- $4 \rightarrow 4 + 8 = 12 \rightarrow 1100$
- $-1 \rightarrow -1 + 8 = 7 \rightarrow 0111$

■ Nella notazione eccesso 8 $\rightarrow 8 = 2^{(n-1)} \rightarrow n=4$

- N. **piu' grande** rappresentabile : $2^{(n-1)} - 1 \rightarrow 7$
- N. **piu' piccolo** rappresentabile: $-2^{(n-1)} \rightarrow -8$



Rappresentazione dell'informazione - 5

- Da **floating point** a decimale

- 0 101 1011

- positivo
 - 0, mantissa : 0,1011
 - Esponente come n. eccesso 4 (su tre bit)
 $101 \rightarrow 5-4=1 \rightarrow$ spostato la virgola di 1 posto a dx
 - 1,011 \rightarrow in decimale **1,375**

- Da **decimale** a floating point