CPU e programmazione (Parte 2)

Esercizio “somma-mag”

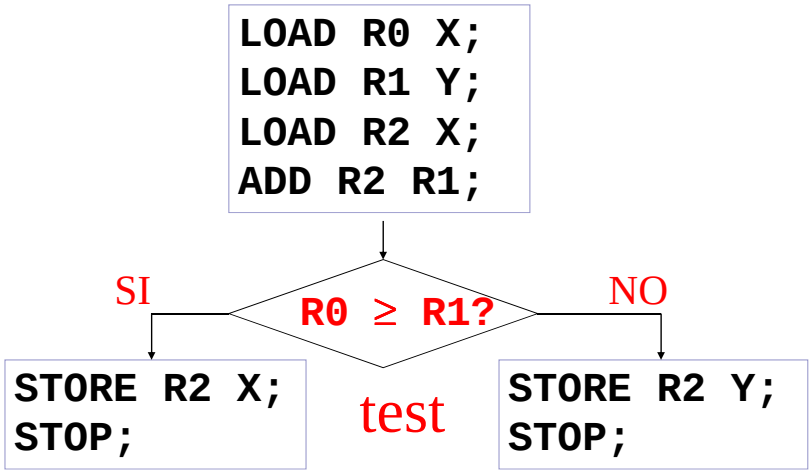
Caricare due valori interi dalla RAM, sommarli e mettere il **risultato** al posto del maggiore dei 2 numeri sommati (nel caso siano uguali, non importa in quale dei due si mette la somma)

```
X: INT 38;
Y: INT 8;
    LOAD R0 X;
    LOAD R1 Y;
    LOAD R2 X;
    ADD R2 R1;
    COMP R0 R1;
    BRGE pippo;
    STORE R2 Y;
    STOP;
pippo: STORE R2 X;
    STOP;
```

Vantaggi dell’Assembler

- Il programma scritto in Assembler rispecchia molto da vicino il programma scritto in linguaggio macchina
- Inoltre è facile riconoscere la sua struttura e rappresentarla con un diagramma di flusso

Flowchart



Esercizio “somma-pos”

Definiti due float il programma deve effettuare la somma dei due valori e porla nel primo, se il primo è strettamente positivo, altrimenti nel secondo.

Esercizio “somma-pos”

```
X: FLOAT 1.0;
Y: FLOAT  3.0;
ZERO: FLOAT 0.0;

LOAD R0 X;
LOAD R1 Y;
LOAD R3 ZERO;
FCOM R0 R3;
FADD R0 R1;
BRGT positivo;
STORE R0 Y;
STOP;
positivo: STORE R0 X;
STOP;
```

Esercizio “mult-pos”

Scrivere un programma che, dati due interi, se entrambi sono strettamente positivi, calcola la moltiplicazione e la pone nel primo, altrimenti calcola la somma e la pone nel primo.

Esercizio “mult-pos”

```
X: INT 1;
Y: INT 3;
ZERO: INT 0;

LOAD R0 X;
LOAD R1 Y;
LOAD R3 ZERO;
COMP R0 R3;
BRLE negativi;
COMP R1 R3;
BRLE negativi;
```

```
MUL R0 R1;
STORE R0 X;
STORE R1 Y;
STOP;
negativi: ADD R0 R1;
STORE R0 X;
STORE R1 Y;
STOP;
```