Novità dell'Assembler

- **codici mnemonici** per le **operazioni**
- **nomi mnemonici** (**identificatori**) **al posto degli indirizzi RAM** per i dati (e indirizzi RAM delle istruzioni usate nei salti)
- tipi dei dati **`INT`** e **`FLOAT`**

Codice-op mnemonici:

- **trasferimento**: **`LOAD`** (RAM → CPU) e **`STORE`** (CPU → RAM)
- **aritmetiche**: **`ADD, SUB, DIV, MULT, MOD, FADD, FSUB, FDIV, FMULT`**
- **input/output**: **`READ`** (U-INP → CPU), **`WRITE`** (CPU → U-OUT)
- **test**: **`COMP, FCOMP`**
- **salto**: **`BREQ, BRGT, BRLT, BRGE, BRLE, BRANCH`**
- **terminazione**: **`STOP`**

Stesso esempio del linguaggio macchina

```
Z : INT ;
X : INT 38;
Y : INT 8;
LOAD R0 X;
LOAD R1 Y;
ADD  R0 R1;
STORE R0 Z;
```

dichiarazioni degli **identificatori dei dati**

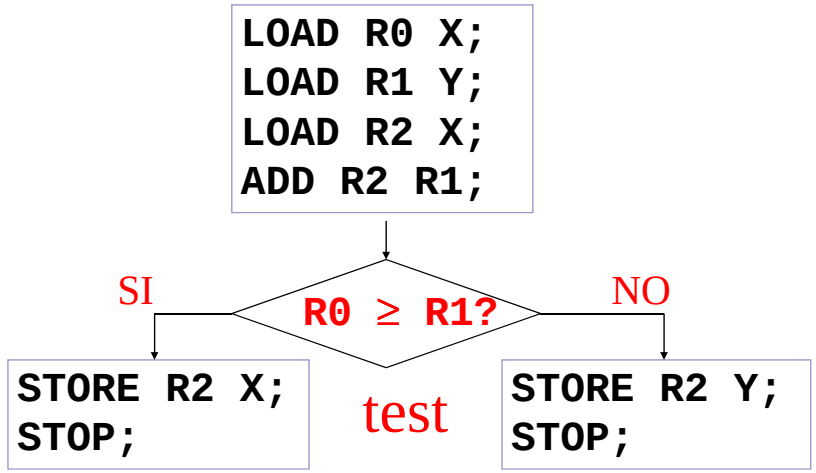
istruzioni assembler

Esempio

carica due valori dalla RAM, li **somma** e mette 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;
```

Flowchart



Esempio

Scrivere un programma che dato un reale X restituisce il valore assoluto |X|. Quindi:

1. legge X dalla tastiera,
2. **se** X ≥ 0 lo scrive direttamente sul monitor **altrimenti**,
3. lo cambia di segno (ad es. sottraendolo a 0) prima di scriverlo sul monitor.

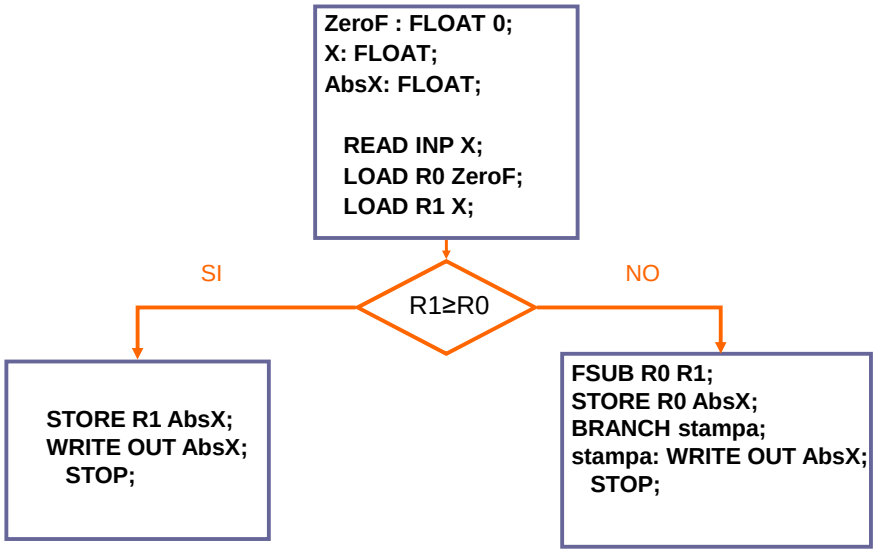
```
ZeroF : FLOAT 0;
X: FLOAT;
AbsX: FLOAT;

  READ INP X;
  LOAD R0 ZeroF;
  LOAD R1 X;
  FCOMP R1 R0;
  BRGE maggiore;
  FSUB R0 R1;
  STORE R0 AbsX;
  BRANCH stampa;

maggiore: STORE R1 AbsX;
   BRANCH stampa;

stampa: WRITE OUT AbsX;
  STOP;
```

FLOWCHART



Esempio

Leggere un reale x ed un intero positivo n e calcolare la potenza x^n

Questo esempio utilizza una struttura di controllo di grande importanza: **il ciclo o iterazione**

```
X: FLOAT ;
N: INT ;
Ris: FLOAT ;
Uno : INT 1;
Unofl: FLOAT 1.0;
READ INP X;
READ INP N;
LOAD R0 Uno;
SUB R0 R0;
LOAD R1 Uno;
LOAD R2 X;
LOAD R3 N;
LOAD R4 Unofl;
```

R0 = 0 intero
R1 = 1 intero
R2 = X reale
R3 = N intero
R4 = 1 reale

R0 = 0 intero
R1 = 1 intero

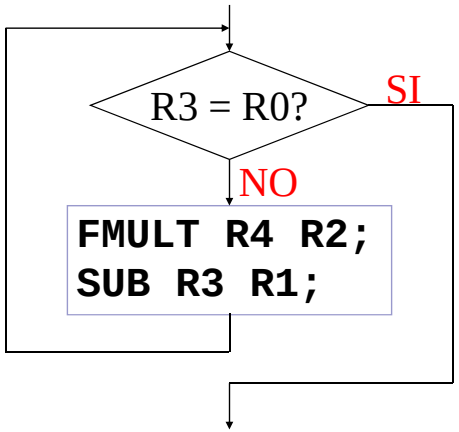
R2 = X reale
R3 = N intero

R4 = 1 reale

```
Ciclo: COMP R3 R0;
  BREQ Esci;
  FMULT R4 R2;
  SUB R3 R1;
  BRANCH Ciclo;
Esci: STORE R4 Ris;
  WRITE STOUT Ris;
  STOP;
```

R4 = X^{N-R3}

R4 = X^N



ciclo o iterazione

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)

Informazioni

- **Non c'è lezione dal 22/10 al 27/10 !!!!**

- Lezioni in laboratorio:
  - Martedi' 6 Novembre 14:30-16:30
  - Martedi' 13 Novembre 14:30-16:30
  - Martedi' 20 Novembre 14:30-16:30
  - Martedi' 27 Novembre 14:30-16:30
  - Martedi' 4 Dicembre 14:30-16:30