

Svantaggi del linguaggio macchina

- programmi in binario sono difficili da scrivere, capire e modificare
- il programmatore deve occuparsi della gestione degli indirizzi RAM

primo passo $\Rightarrow$ **Assembler**

Caratteristiche dell'Assembler

L'Assembler e' un rudimentale linguaggio di programmazione

- codici mnemonici per le operazioni
- nomi mnemonici, detti *variabili*, al posto degli indirizzi RAM dei dati
- nomi mnemonici, detti *etichette*, al posto degli indirizzi RAM delle istruzioni usati nei salti
- categorie di valori: si tratta dei cosiddetti *tipi di dato* **INT** e **FLOAT** (normalmente non presenti)

Codici mnemonici di operazioni

- **trasferimento**: **LOAD** (RAM → CPU) e **STORE** (CPU → RAM)
- **aritmetiche**: **ADD**, **SUB**, **MULT**, **DIV**, **MOD**, **FADD**, **FSUB**, **FMULT**, **FDIV**
- **test**: **COMP**, **FCOMP**
- **salto**: **BREQ**, **BRGT**, **BRLT**, **BRGE**, **BRLE**, **BRANCH**
- **terminazione**: **STOP**

Esempio precedente in Assembler

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

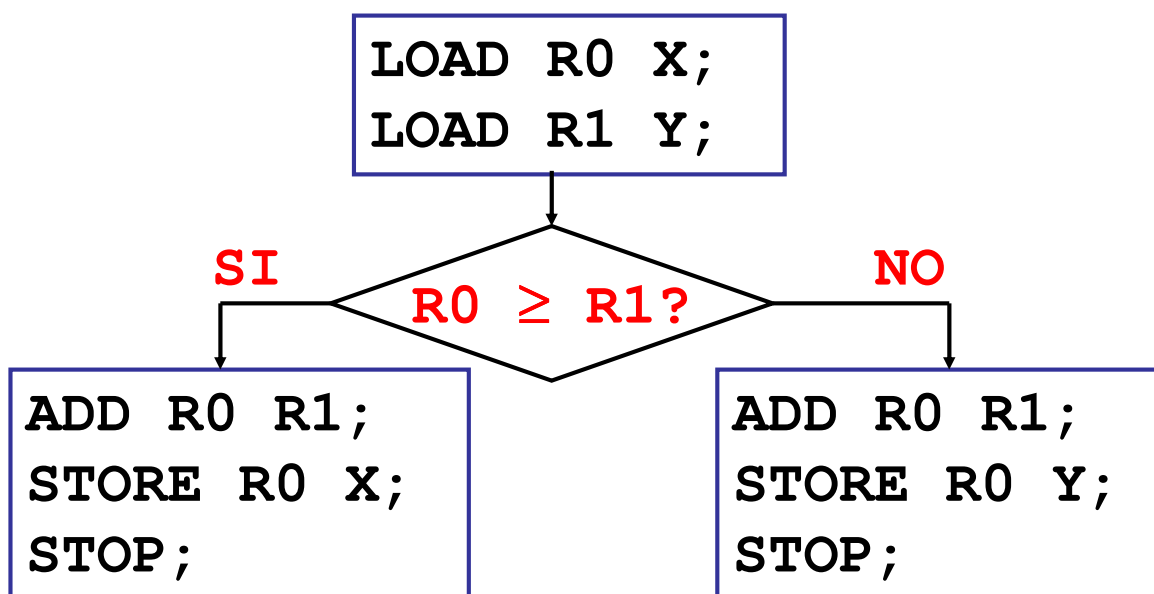
dichiarazioni di
variabili

istruzioni

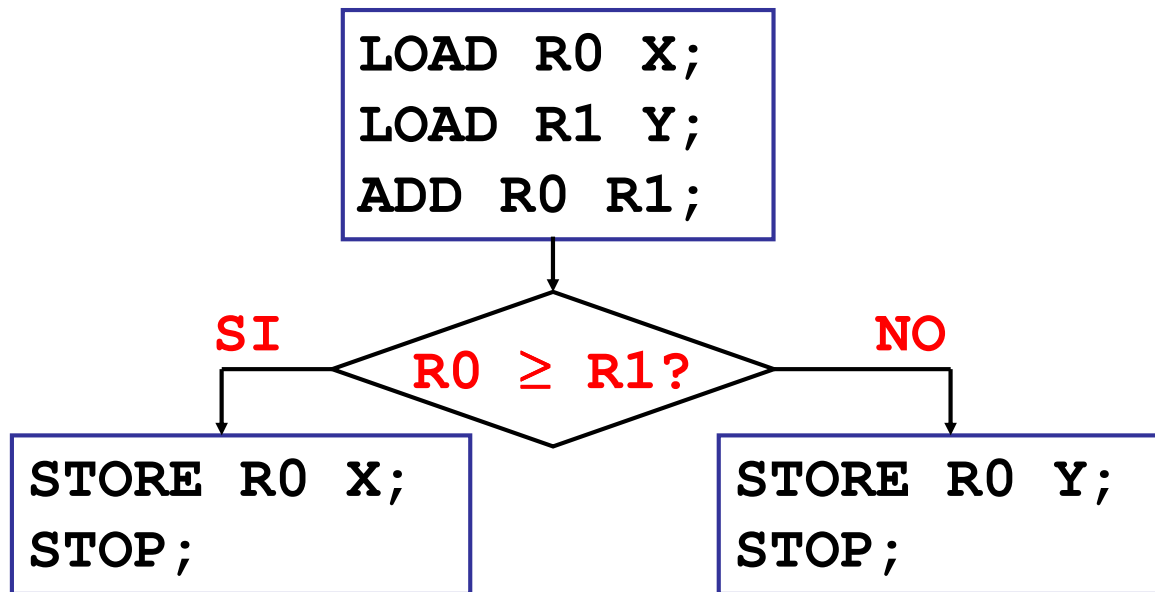
Altro esempio

Carica due valori dalla RAM, li somma e memorizza il risultato al posto del maggiore dei 2 numeri sommati (nel caso siano uguali, in uno qualsiasi dei 2 indirizzi RAM)

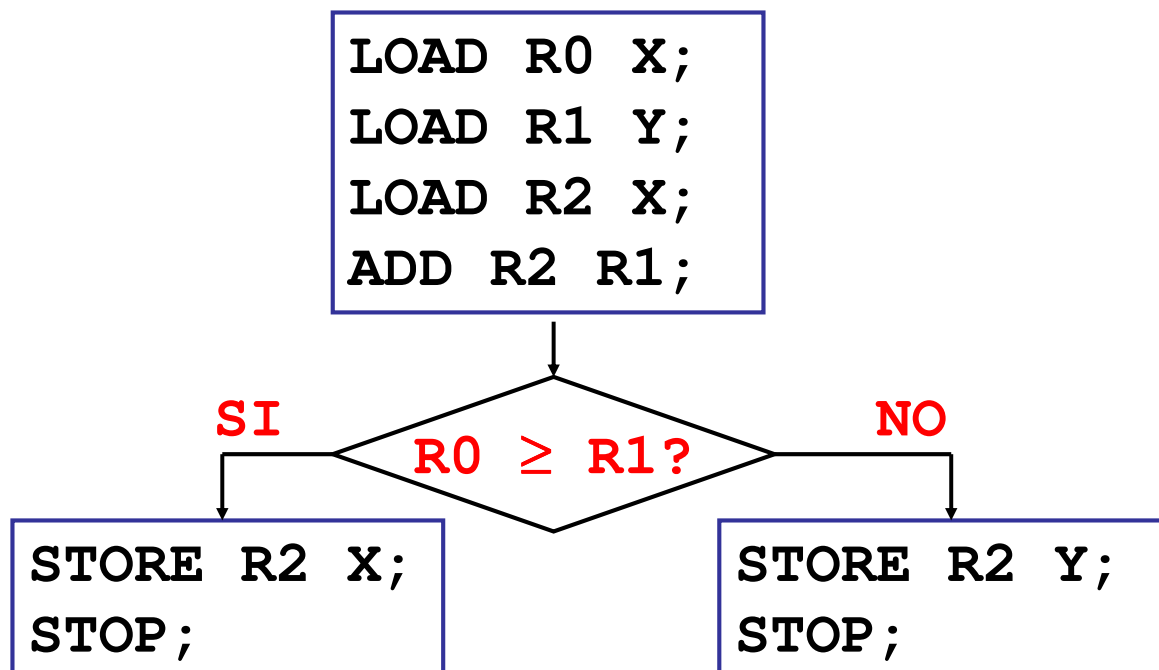
Algoritmo 1



Questo algoritmo funziona?



Algoritmo 2



Codice assembler algoritmo 1

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

Codice assembler algoritmo 2

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

Conclusioni

3 costrutti fondamentali:

1) $X \leftarrow \text{valore}$

ASSEGNAZIONE

2) **TEST** di una condizione che determina quale azione intraprendere successivamente

3) **CICLO** che si ripete finchè una data condizione è vera

SONO COSTRUTTI FONDAMENTALI in ogni LINGUAGGIO di PROGRAMMAZIONE

La CPU non "capisce" l'assembler !!

Il programma assembler deve essere tradotto in un programma macchina

