La toolchain di C

Alcune domande ovvie

- Le macchine reali sono costruite sullo schema di von Neumann:
  - Chi si occupa di "colmare il gap" tra la macchina astratta C e la macchina reale (del tipo di von Neumann)?
  - Risposta: il "software di base"
  - Più precisamente
    - il **compilatore**, o, più raramente,
    - l'**interprete**
    - il **linker**
- Nulla di magico neanche nella macchina C
  - Diamo una breve occhiata al compito del compilatore

Principali funzioni del compilatore

- Dal simbolico al binario
- Dall'aritmetica infissa all'aritmetica della ALU (Arithmethic Logic Unit)
- Dalle istruzioni composte al program counter
- La gestione della memoria
- ….

Dal simbolico al binario

LOAD	00110
STORE	00111
BR	00001
…	….

A	100000
B	100001
x	100010
alfa	100011
….	
ciclo1	110000
etichetta	110100
…	…

Dall'aritmetica infissa all'aritmetica della ALU

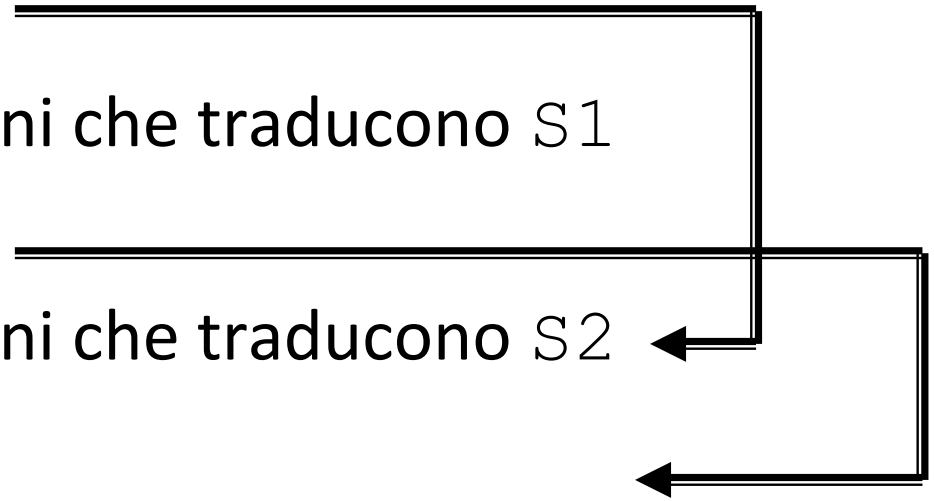
- (a+b)*(c+d) ---->
- LOAD A
- ADD B
- STORE TEMP
- LOAD C
- ADD D
- MULT TEMP

- …
- Algoritmi di traduzione non del tutto banali

Istruzioni composte: if … else

- **`if`** `(cond) S1` **`else`** `S2;`

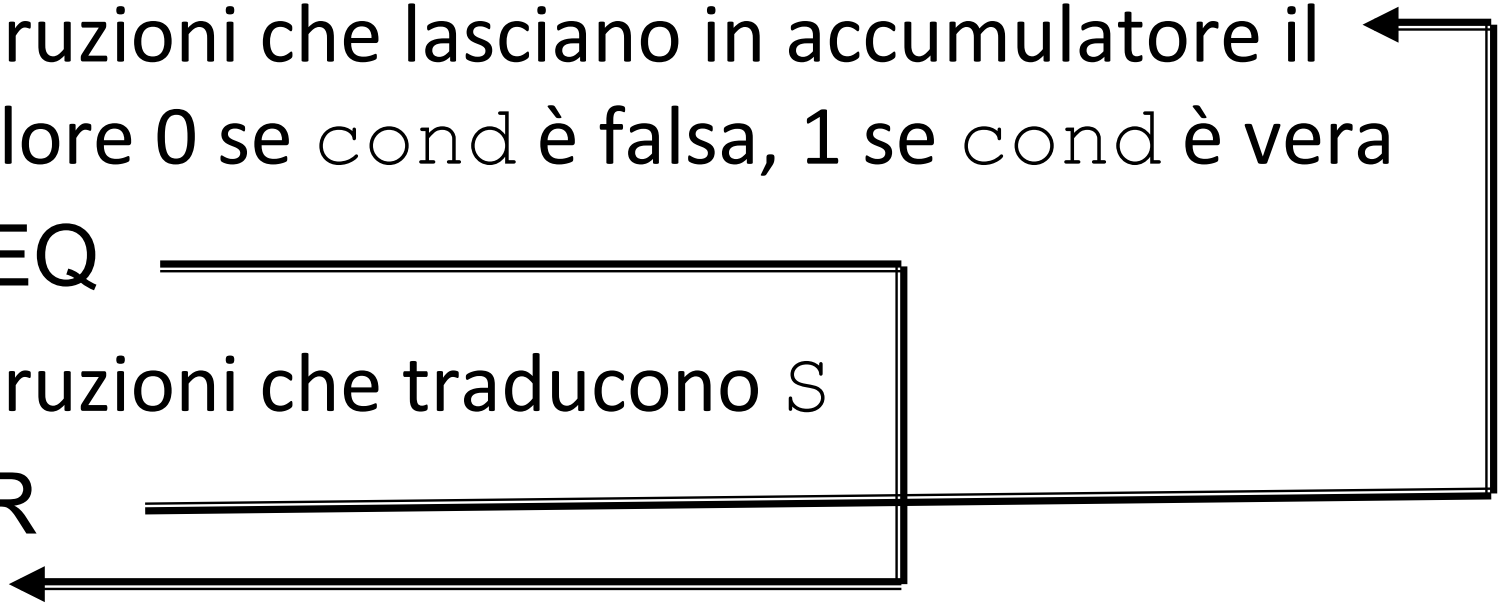
- Istruzioni che lasciano in accumulatore il valore 0 se `cond` è falsa, 1 se `cond` è vera
- BEQ
- Istruzioni che traducono `S1`
- BR
- Istruzioni che traducono `S2`



Istruzioni composte: while

- **`while`** `(cond) S;`

- Istruzioni che lasciano in accumulatore il valore 0 se `cond` è falsa, 1 se `cond` è vera
- BEQ
- Istruzioni che traducono `S`
- BR



L’interprete

- Invece che tradurre il linguaggio di alto livello in codice macchina, le istruzioni possono essere “simulate”
- L’interprete altro non è che un “simulatore” di una macchina che esegue, istruzione per istruzione, programmi scritti in determinati linguaggi
- Vantaggi (tra i tanti): più semplice e pratico creare e modificare programmi
- Svantaggi (tra i tanti): esecuzione più lenta, minori controlli di correttezza

Il linker

- Il compilatore genera **file oggetto**
  - Codifiche binarie di istruzioni e dati
- Un file oggetto non è necessariamente direttamente eseguibile
  - Può rappresentare porzioni di codice **riusabili** in diverse applicazioni
- Compito del linker è assemblare diversi file oggetto fino ad ottenere un programma completo e quindi eseguibile
  - Vedremo un esempio concreto quando parleremo di procedure e funzioni