



- Piccolo riepilogo
- Il linguaggio macchina
- Assembly
- Linguaggi di Alto Livello
- Linguaggi Interpretati e Compilati

- Dove si trova il programma da eseguire?
  - In memoria → “numeri”
- La control unit esegue i programmi ovvero:
  - Legge da memoria la relativa istruzione da eseguire → **Fetch**
  - Decodifica il suo significato ovvero cosa deve fare → **Decode**
  - Esegue quanto richiesto pilotando gli altri sistemi → **Execute**
  - Scrive eventuali risultati elaborazione → **Store**
  - Ripete...

- Come è fatta una istruzione decodificata dall'unità di controllo?
- Tipicamente più “parti”
  - Operation code → mi dice che istruzione è
  - Operando/i → mi fa capire su cosa si opera
- Instruction Set → dipende da processore!
- Esempio: ADD X, Y
  - Somma il contenuto della memoria all'indirizzo X a quello in Y e scrivi il risultato in X

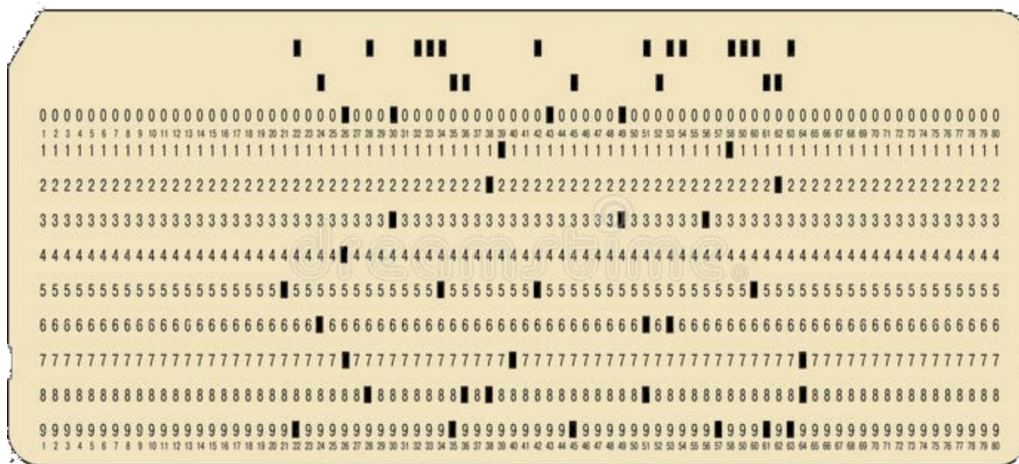
Opcode (8 bit)	Indirizzo X (16 bit)	Indirizzo Y (16 bit)
00001001	0000000001100011	0000000001100100

- Codice numerico che corrisponde alle possibili istruzioni eseguibili da una CPU
- Diverse per ogni tipo di processore
  - Instruction Set
- Ad esempio l'Instruction set dei processori ARM è completamente differente da quello dei processori Intel x86
  - Differenti modelli della stessa famiglia di processori presentano Instruction Set con variazioni

## 6502 Machine Language Hex Codes

[illegible]

- Non mnemonico → difficoltà di utilizzo
- Non portabile
  - Codice sviluppato per una famiglia di CPU non utilizzabile altrove
- All'atto pratico non usato direttamente



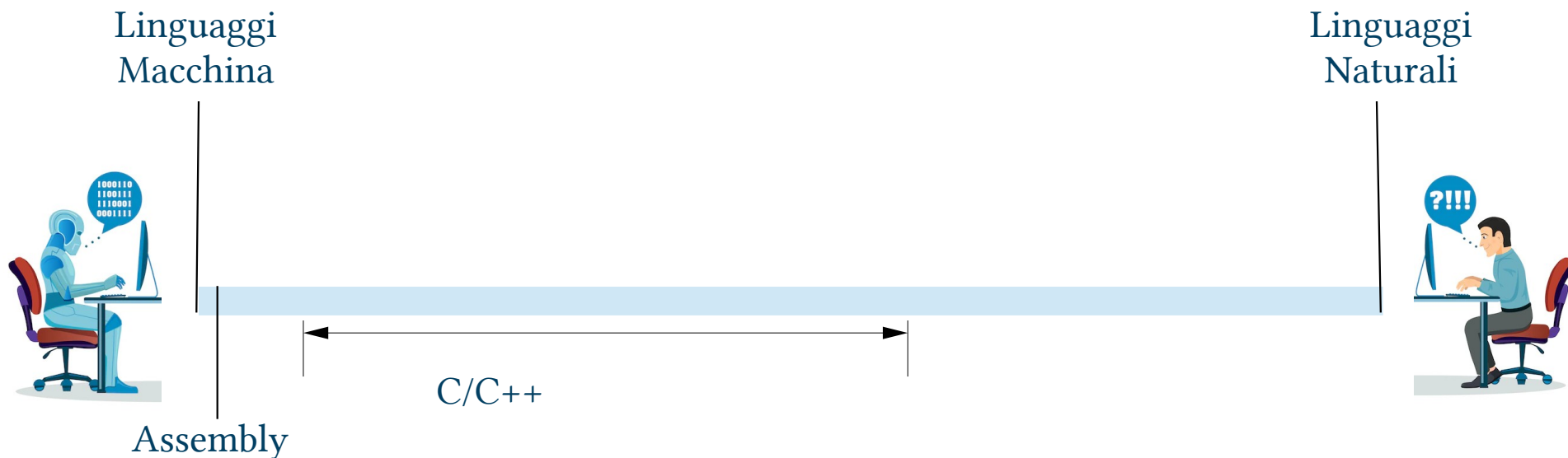
- Sostituzione codice numerico con simboli mnemonici
- ~1:1

```
START: MOV    AX, BX
        CMP    AX, 12h
        JZ     EQUAL
        INT    21h
        RET
EQUAL:  MOV    BL, 82h
```



- Molto più facile del codice macchina!
  - Sia da scrivere che da leggere
- Ma devo comunque ottenere il codice macchina
- Assembler: programma che legge le istruzioni assembly e le traduce in istruzioni macchina
  - Quasi traduzione 1 a 1
- Comunque linguaggio complesso
  - Molte istruzioni anche per operazioni semplici
- Non portabile

- Non più corrispondenza 1:1 con istruzioni linguaggio macchina
  - 1 istruzione può corrispondere a decine di istruzioni linguaggio macchina



# Codice Macchina

[illegible]

# Assembly

```
.file      "hello.asm"
.text
.section   .rodata

.LC0:
.string "Hello World"
.text
.globl     hello
.type      hello, @function

main:
.LFB0:
.cfi_startproc
endbr64
pushq     %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq      %rsp, %rbp
.cfi_def_cfa_register 6
subq      $16, %rsp
movl      %edi, -4(%rbp)
movq      %rsi, -16(%rbp)
leaq      .LC0(%rip), %rdi
call      puts@PLT
movl      $0, %eax
leave
.cfi_def_cfa 7, 8
ret
.cfi_endproc

.LFE0:
```

# FORTRAN

```
program hello
  print *, 'Hello, World!'
end program hello
```

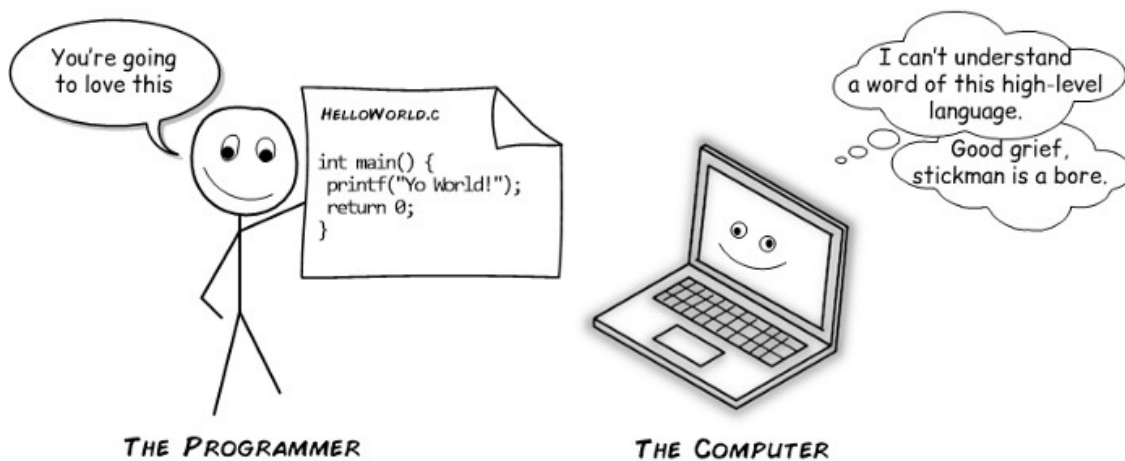
# Elenco (ampiamente incompleto!)

- FORTRAN (1954)
- LISP (1958)
- ALGOL (1958) & ALGOL68 (1968)
- PASCAL (1970)
- C (1969)
- C++ (1979)
- JAVA (1991)
- Python (1991)

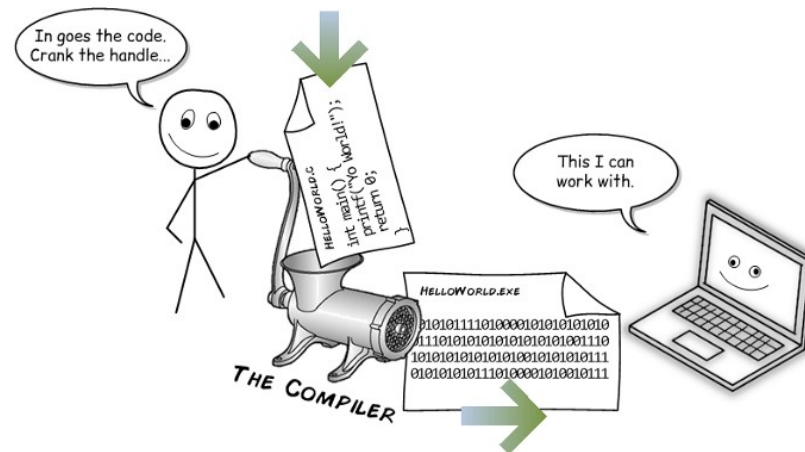


- Pro
  - Mascherano il calcolatore
  - Maggiore leggibilità e comprensibilità
  - Portabilità
- Contro
  - Necessità di traduzione

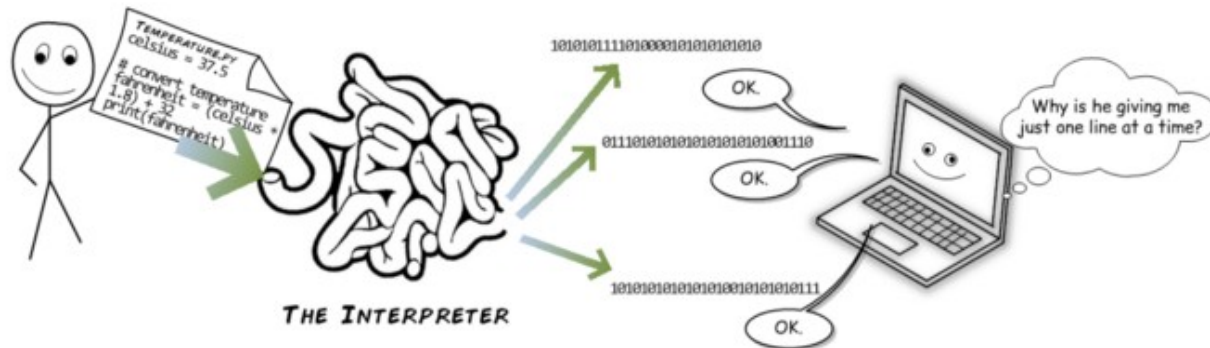
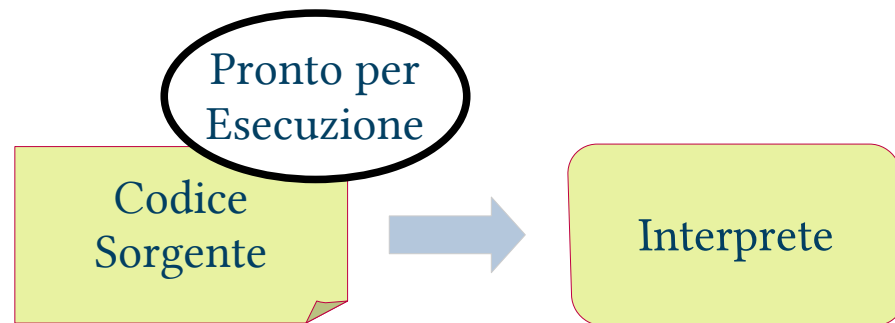
- I linguaggi ad alto livello richiedono una traduzione più complessa
  - Da codice sorgente a linguaggio macchina
- Due possibilità
  - Compilazione
  - Interpretazione



- Traduzione in codice macchina necessaria
- Compilatore → il “traduttore”
- Produce un “file eseguibile”



- Il codice sorgente non deve essere tradotto
- Interprete che lo “legge” al volo
  - Macchina virtuale





- Compilazione
  - Più veloce
  - Permette progetti multifile con più linguaggi
  - Posso nascondere codice sorgente
- Interpretazione
  - Esecuzione immediata
    - Scripting
  - Multiplatforma
    - Web e altro

# Perché così tanti linguaggi?

- Ricerca costante di sistemi migliori per aggiornamento tecnologia
- Non solo “General Purpose” ma problemi specifici
  - Web → Javascript, PHP, GO...
  - Mobile → JAVA
  - Firmware → C
  - Sistemi real time → C++
- Performance vs Rapidità di sviluppo
  - GO/C ↔ Javascript/Python

- Lessico
  - Parole chiave: istruzioni, operatori, simboli...
- Sintassi
  - Come si possono combinare i precedenti elementi
    - Parsing
  - Si può distinguere tra
    - Sintassi necessaria e
    - ~~Zucchero Sintattico~~

**Errore di  
Compilazione o  
Interpretazione**

- Semantica
  - ~~Significato di quanto scritto~~

**Errore di  
Esecuzione**

- Computer: sistema che esegue una serie di istruzioni in “sequenza” → automa
- Programmare:
  - Individuare algoritmo risolutivo del problema dato
    - Passi e loro ordine di esecuzione
  - Tradurre l'algoritmo individuato in codice
    - Lessico, Sintassi e Semantica specifici del linguaggio di programmazione

