

# I Linguaggi di Programmazione

There will always be things we wish to say in our programs that in all languages can only be said poorly

### Sommario



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

## Riepilogo: Unità di Controllo

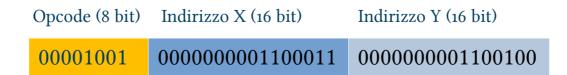


- Dove si trova il programma da eseguire?
  - In memoria → "numeri"
- La control unit esegue i programmi ovvero:
  - Legge da memoria la relativa istruzione da eseguire  $\rightarrow$  **Fetch**
  - Decodifica il suo significato ovvero cosa deve fare  $\rightarrow$  **Decode**
  - Esegue quanto richiesto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

## Riepilogo: Esempio Istruzione



- Come è fatta una istruzione decodificata dall'unità di controllo?
- Tipicamente piú "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



## Linguaggio Macchina



- Codice <u>numerico</u> 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

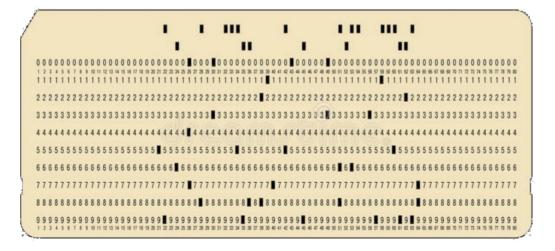
### 6502 Machine Language Hex Codes

	Immediate	Accumulator	<sup>Zero</sup> Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Indirect	(Zero Page, X)	(Zero Page), y	Implied	Relative	Affects Flags
			Zer.											
Add with Carry	69		65	75		6D	7D	79		61	71			N,V,Z,C
Bitwise AND with Accumulator	29		25	35		2D	3D	39		21	31			N,Z
Arithmetic Shift Left		0A	06	16		0E	1E							N,Z,C
Branch on Carry Clear													90	
Branch on Carry Set													B0	
Branch on Equal													F0	
Test Bits			24			2C								N,V,Z
Branch on Minus													30	
Branch on Not Equal													D0	
Branch on Plus													10	
Break												00		В
Branch on Overflow Clear													50	
Branch on Overflow Set													70	
Clear Carry												18		С
Clear Decimal												D8		D
Clear Interrupt												58		1
Clear Overflow												B8		V
Compare Accumulator	C9		C5	D5		CD	DD	D9		C1	D1			N,Z,C
Compare X Register	E0		E4			EC								N,Z,C
Compare Y Register	C0		C4			CC								N,Z,C
Decrement Memory			C6	D6		CE	DE							N,Z
Decrement X												CA		N,Z
Decrement Y												88		N,Z
Bitwise Exclusive OR	49		45	55		4D	5D	59		41	51	- 00		N,Z
Increment Memory	10		E6	F6		EE	FE	- 00		-12	01			N,Z
Increment X												E8		N,Z
Increment Y												C8		N,Z
Jump						4C			6C			- 00		14,2
Jump to Subroutine						20			00					
Load Accumulator	A9		A5	B5		AD	BD	В9		A1	B1			N,Z
Load X Register	A2		A6	B6		AE		BE		7.1				N,Z
Load Y Register	A0		A4	B4		AC	вс	DL						N,Z
Logical Shift Right	7.0	4A	46	56		4E	5E							N,Z,C
No Operation		7/1	40	50		7.	JL					EA		14,2,0
Bitwise OR with Accumulator	09		05	15		0D	1D	19		01	11	LA		N,Z
Push Accumulator to Stack	09		05	13		UD	10	19		OI	11	48		14,2
Push Processor Status to Stack												08		
Pull Accumulator off Stack												68		
Pull Processor Status off Stack												28		All
Rotate Left		2A	26	36		2E	3E					20		N,Z,C
Rotate Right		6A	66	76		6E	7E							N,Z,C
Return from Interrupt		0A	00	70		OE	/ =					40		All
Return from Subroutine												60		All
	E9		E5	F5		ED	FD	F9		E1	F1	60		NVZC
Subtract with Carry	E9		ES	F5		ED	FD	F9		EI	LI	20		N,V,Z,C
Set Carry Set Decimal												38 F8		C D
Set Interrupt			05	05		0.0	0.0	00		04	04	78		1
Store Accumulator			85	95	00	8D	9D	99		81	91			
Store X Register			86	0.4	96	8E								
Store Y Register			84	94		8C								
Transfer A to X												AA		N,Z
Transfer A to Y												A8		N,Z
Transfer Stack Pointer to X												BA		
Transfer X to A												8A		N,Z
Transfer X to Stack Pointer												9A		
Transfer Y to A												98		N,Z

## Linguaggio Macchina



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



## Assembly



• Sostituzione codice numerico con simboli mnemonici

• ~1:1

```
START: MOV AX, BX
CMP AX, 12h
JZ EQUAL
```

INT 21h

**RET** 

EQUAL: MOV BL, 82h

## **Assembly**



- Molto piú 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

## Linguaggi ad Alto Livello



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



#### Codice Macchina

#### 0000f30f1efa4883ec08488d3d00000000e8000000031c04883c408c3004743433a20285562756e747520392e342e302d317562756e7475317e32302e30342920392e342e30000000000400000000017810011b0c070890010000140000001c00000000000001b0000000480e10520e08000000742e73746172747570002e636f6d6d656e74002e6e6f74652e474e552d737461636b002e6e6f74

### Assembly

```
.file
                "hello asm"
        .text
        .section
                         . rodata
.LC0:
        .string "Hello World"
        .text
        .globl
                hello
        .tvpe
                hello, @function
main:
.LFB0:
        .cfi startproc
        endbr64
        pusha %rbp
        .cfi def cfa offset 16
        .cfi offset 6. -16
                %rsp, %rbp
        .cfi def cfa register 6
        subq
                $16, %rsp
        mov1
                %edi, -4(%rbp)
        mova
                %rsi, -16(%rbp)
        lead
                .LCO(%rip), %rdi
        ca 11
                puts@PLT
        mov1
                $0, %eax
        1eave
        .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)



## Caratteristiche Linguaggi ad Alto Livello



### • Pro

- Mascherano il calcolatore
- Maggiore leggibilità e comprensibilità
- Portabilità

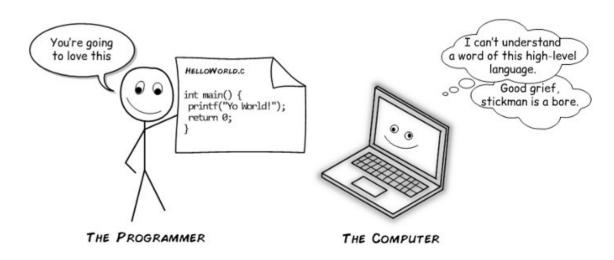
### Contro

Necessità di traduzione

## Linguaggi interpretati e compilati



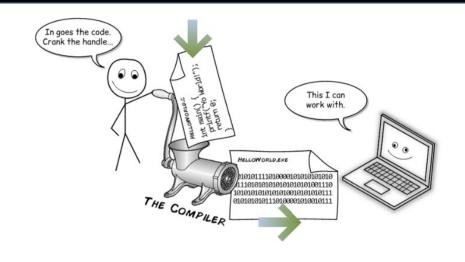
- I linguaggi ad alto livello richiedono una traduzione piú complessa
  - Da codice sorgente a linguaggio macchina
- Due possibilità
  - Compilazione
  - Interpretazione

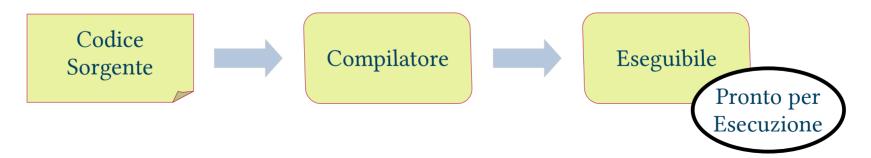


## Compiled Language



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



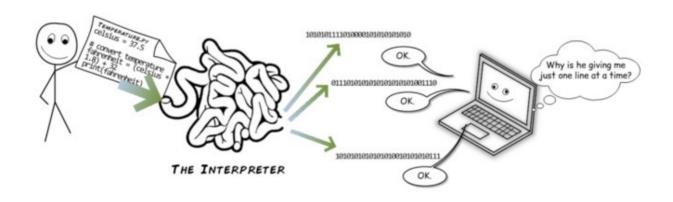


## **Interpreted Language**



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





## **Compiled vs Interpreted**



- Compilazione
  - Piú veloce
  - Permette progetti multifile con piú linguaggi
  - Posso nascondere codice sorgente
- Interpretazione
  - Esecuzione immediata
    - Scripting
  - Multipiattaforma
    - Web e altro

## Perché così tanti linguaggi?



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

## Elementi del linguaggio



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

Errore di Compilazione o Interpretazione

Errore di Esecuzione

## Riepilogo



- 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



# I Linguaggi di Programmazione



There will always be things we wish to say in our programs that in all languages can only be said poorly

Alan Perlis