Programmazione

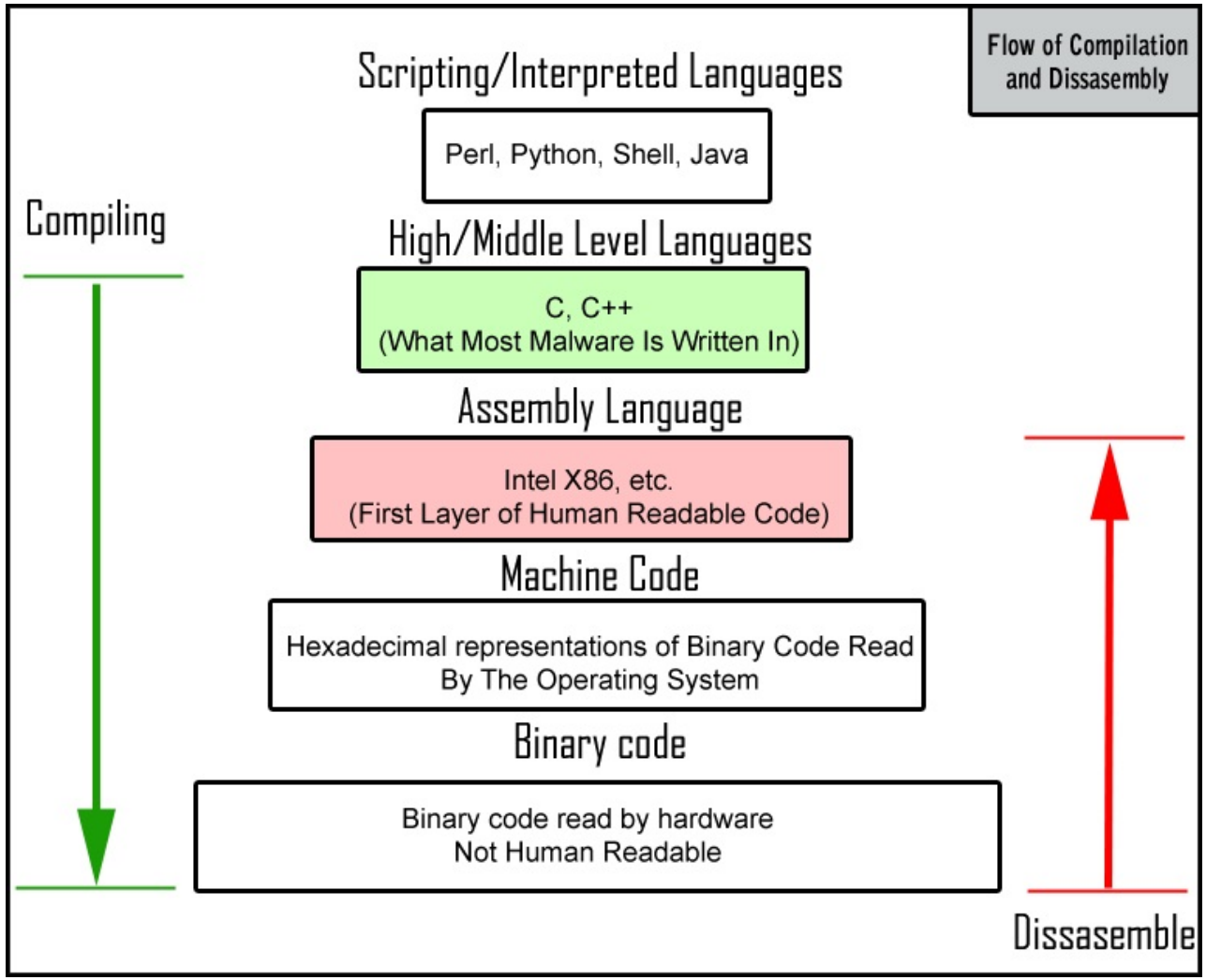
Dipartimento di Matematica

Ing. Cristiano Gregnanin

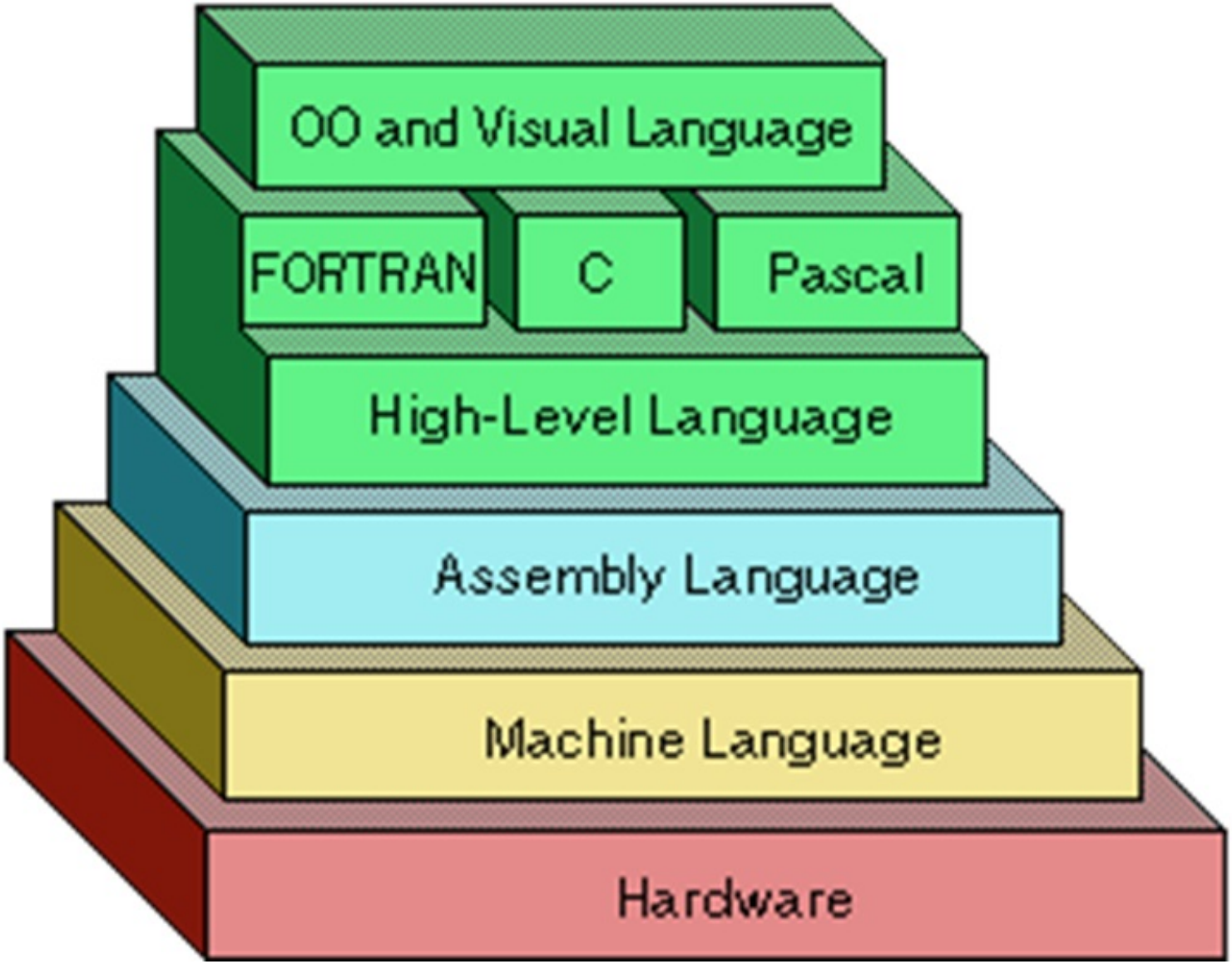
Corso di laurea in Matematica

29 febbraio 2016

Linguaggi


Flow of Compilation and Dissasembly
Scripting/Interpreted Languages
Perl, Python, Shell, Java
Compiling
High/Middle Level Languages
C, C++
(What Most Malware Is Written In)
Assembly Language
Intel X86, etc.
(First Layer of Human Readable Code)
Machine Code
Hexadecimal representations of Binary Code Read By The Operating System
Binary code
Binary code read by hardware
Not Human Readable
Dissasemble

Linguaggi


OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware

Linguaggi di alto livello

Si basano su una **macchina virtuale** le cui *mosse* non sono quelle della macchina hardware

Cos è un linguaggio?

è un **insieme di parole** e di **metodi di combinazione delle parole** *usati e compresi da una comunità di persone.*
è una definizione poco precisa:

- non evita le ambiguità dei linguaggi di naturali.
- non si presta a descrivere processi computazionali meccanizzabili.
- non aiuta a stabilirne le proprietà

Nozione di linguaggio

Occorre una nozione di linguaggio più precisa: linguaggio come sistema matematico che consenta di rispondere ad alcune domande:

- quali sono le frasi lecite?
- si può stabilire se una frase appartiene al linguaggio?
- come si stabilisce il significato di una frase?
- quali elementi linguistici primitivi?

Esempio: espressioni

- Vogliamo rappresentare le espressioni che si possono scrivere con le 4 operazioni ed i numeri naturali.
- Serve un modo per dire che $1 + 2 * 3$ è lecito mentre $1 + *23$ non lo è.
- Il significato potrebbe essere il risultato: il significato di $1 + 2 * 3$ è 7
- I numeri interi e le 4 operazioni

Linguaggio e programma

- Dato un algoritmo, **un programma** è la sua descrizione in un particolare linguaggio di programmazione
- Un linguaggio di programmazione è una notazione formale che può essere usata per descrivere algoritmi. I linguaggi sono caratterizzati da almeno 2 aspetti: **sintassi** e **semantica**

Sintassi e semantica

- **Sintassi: l'insieme di regole formali per la scrittura di programmi** in un linguaggio, che dettano le **modalità per costruire frasi corrette** nel linguaggio stesso
- **Semantica:** l'insieme dei significati da attribuire alle frasi (*sintatticamente corrette*) costruite nel linguaggio.

Osservazione: la frase può essere sintatticamente corretta eppure non avere significato.

Sintassi

Le regole sintattiche sono espresse attraverso **notazioni formali:**

- **BNF** Backus-Naur Form
- **EBNF** Extended Backus-Naur Form
- Diagrammi sintattici

Sintassi EBNF: esempio

$$< naturale >::= 0| < cifra - non - nulla > [< cifra >]$$
$$< cifra - non - nulla >::= 1|2|...|9$$
$$< cifra >::= 0| < cifra - non - nulla >$$

Sintassi di un numero naturale

$< naturale >::= 0| < cifra - non - nulla > [< cifra >]$
Un naturale si può scrivere come **0 oppure come una cifra non nulla** seguita da zero o più cifre.
$< cifra - non - nulla >::= 1|2|...|9$
Una cifra non nulla si può scrivere come 1 oppure 2 oppure 3...
$< cifra >::= 0| < cifra - non - nulla >$
Una cifra si può riscrivere come 0 oppure come una cifra non nulla

Semantica

La semantica è esprimibile:

- a parole. (poco precisa ed ambigua)
- mediante azioni: **semantica operazionale**
- mediante funzioni matematiche: **semantica denotazionale**
- mediante formule logiche: **semantica assiomatica**

Astrazione

Esistono linguaggi a vari livelli di astrazione:

- Linguaggio macchina: implica la conoscenza dei metodi di rappresentazione delle informazioni utilizzati.
- Linguaggio macchina e assembler: implica la conoscenza dettagliata delle caratteristiche della macchina (registri, dimensioni dati, set di istruzioni). **semplici** algoritmi implicano la specifica di molte istruzioni.
- **linguaggi di alto livello:** Il programma può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico. **Sono indipendenti dalla macchina hardware sottostante. La macchina hardware si dice che è astratta**

Astrazione

- **Linguaggio macchina:**
  0100 0000 0000 1000
  Difficile da leggere e scrivere.
- **Linguaggio macchina e assembler:**
  LOADA H
  LOADB Z
  ADD
  ..
  Le istruzioni corrispondono univocamente a quelle macchina, ma vengono espresse tramite nomi simbolici (parole chiave).
- **linguaggi di alto livello:**
  print('hello world')
  Sono indipendenti dalla macchina hardware.

Realizzazione

Il calcolatore capisce esclusivamente il linguaggio macchina. Per utilizzare un linguaggio di alto livello occorre scrivere il programma con un editor di testo (ad esempio Geany) e salvarlo in un file. Tale file prende il nome di **file sorgente**

Realizzazione

Successivamente bisogna fare in modo che il calcolatore capisca ciò che è stato scritto nel **file sorgente**. Esistono **2 modi** per farlo:

- attraverso **l'interpretazione** del programma
- attraverso **la compilazione** del programma

Interprete vs compilatore

L'interprete *è un programma* che legge il file sorgente e lo **esegue** istruzione per istruzione.
finché non è finito il file sorgente:

1. **leggi** un'istruzione dal file sorgente
2. **traduci** l'istruzione in linguaggio macchina
3. **esegui** l'istruzione tradotta. (esecuzione di codice binario)

Interprete vs compilatore

Il compilatore *è un programma* che legge il file sorgente e lo **traduce** in linguaggio macchina, *salvandolo* in un altro file detto **compilato**

finché non è finito il file sorgente:

1. **leggi** un'istruzione dal file sorgente
2. **traduci** l'istruzione in linguaggio macchina
3. **SALVA** l'istruzione tradotta nel file compilato.
   Alla fine del processo si otterrà un file detto **compilato**

Esecuzione di un programma

Per eseguire sulla macchina hardware un programma scritto in un linguaggio di alto livello è necessario tradurre il programma in sequenze di istruzioni di basso livello, direttamente eseguite dal processore attraverso:

- interpretazione (Basic, php, python, matlab...)
- compilazione (C, C++...)

Come sviluppare un programma

Qualunque sia il linguaggio di programmazione scelto occorre scrivere il testo del programma e memorizzarlo in un file. (*fase di sviluppo.*)
Successivamente se il linguaggio è **compilato** allora bisogna compilare il file sorgente ed eseguire il file *compilato*. **Mentre** se il linguaggio è **interpretato** è sufficiente usare l'interprete per eseguire il programma.

Ambienti di programmazione

è l'insieme dei programmi che consentono la scrittura, la verifica e l'esecuzione di nuovi programmi (fasi di sviluppo).
Sviluppo di un programma: Affinché un programma scritto in un qualsiasi linguaggio di programmazione sia comprensibile (e quindi eseguibile) da un calcolatore occorre tradurlo dal linguaggio originario al linguaggio della macchina. Tale operazione è svolta da speciali programmi detti **traduttori**

Traduzione di un programma

Il **traduttore** converte il testo del file sorgente nella corrispondente rappresentazione in linguaggio macchina. (programma eseguibile)

Sviluppo di programmi

I **traduttori** possono operare traducendo ed eseguendo immediatamente ogni singola istruzione del file sorgente(*interpretati*), **oppure** possono tradurre l'intero programma (*senza eseguirlo*) e producono il programma convertito in linguaggio macchina.
Osservazione: *in generale l'esecuzione di programmi compilati è più rapida.*

riassumendo: compilato vs interpretato

I compilatori traducono automaticamente un programma dal linguaggio L a quello macchina
Gli **interpreti** sono programmi capaci di eseguire direttamente un programma in un linguaggio L istruzione per istruzione.

Ambienti di programmazione

riassumento: compilazione

Il compilatore opera la traduzione di un programma sorgente in un **programma oggetto** direttamente eseguibile dal calcolatore.
PRIMA si traduce **TUTTO** il programma **POI** si esegue la versione tradotta.
A volte il programma oggetto è composto da più moduli, dove ogni modulo è un programma compilato. Il *linker* provvede a collegarli fornendo un unico programma eseguibile.

Ambienti di programmazione

Riassumendo: interpretazione

L'interprete traduce ed esegue direttamente ciascuna istruzione del file sorgente, istruzione per istruzione. E' alternativo al compilatore. La traduzione e l'esecuzione sono due fasi immediatamente consecutive.

Compilato vs interpretato