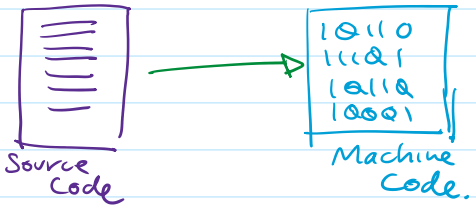
2 Programming Language Implementation

Monday, August 26, 2024 12:05 PM

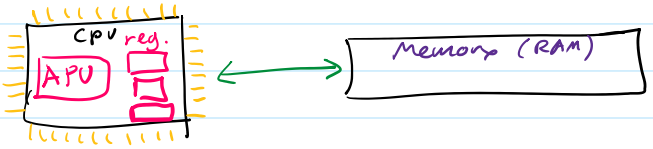
¿How are programming languages implemented?

Objective:



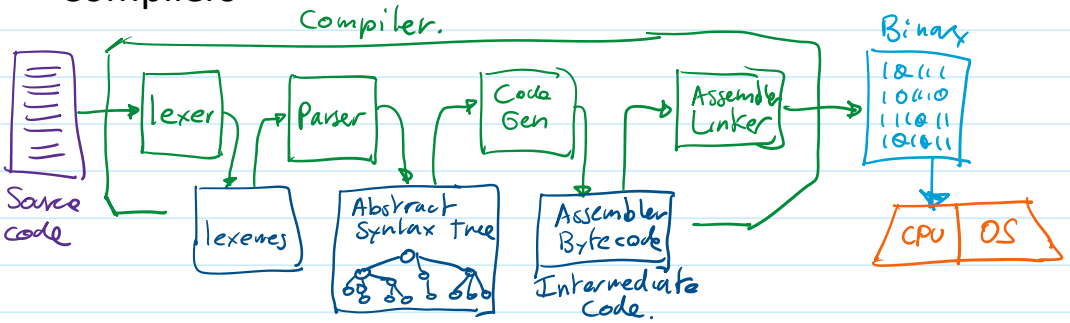
3 Basic types
- Compilers
- Interpreters
- Hybrid (combination of Both)

Note: Machine code is "relatively" simple.



1.- perform operations on values on registers + − ÷ *
2.- fetch data from memory
3.- store data in memory
4.- move, change or modify the "instruction pointer" (to skip or repeat instructions)

- Compilers



- lexer:- splits texts into atomic components
  E.g. apple = banana * (orange + 7.3) ;
- Parser:- recognize whether the lexemes form a valid program under the rules of the language.

- Parser: recognize whether the lexemes form a valid program under the rules of the language

Eg apple | = | 3dfx | [| * | orange | + | (|) | 7 |]

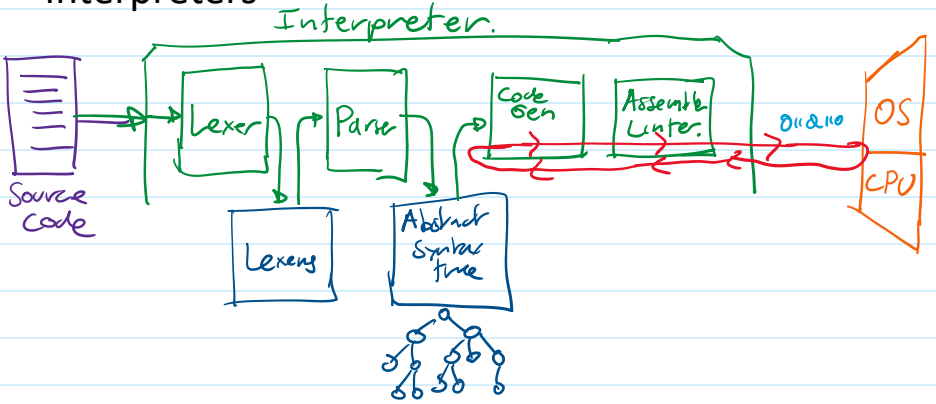
Compilers:

FORTRAN, C, C++, Pascal, D, Go, Rust

Pro: - Speed of final program.
- Some I.P. protection

Cons: - Portability
- Speed of development.
↳ compilation speed can be slow.

- Interpreters



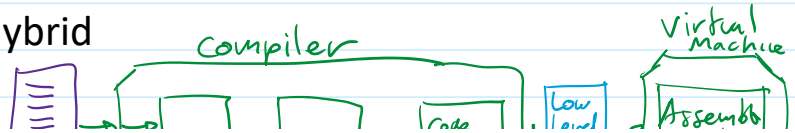
REPL: Read-Evaluate-Print Loop

Eg: LISP, Python, JavaScript, Ruby, ML, BASIC

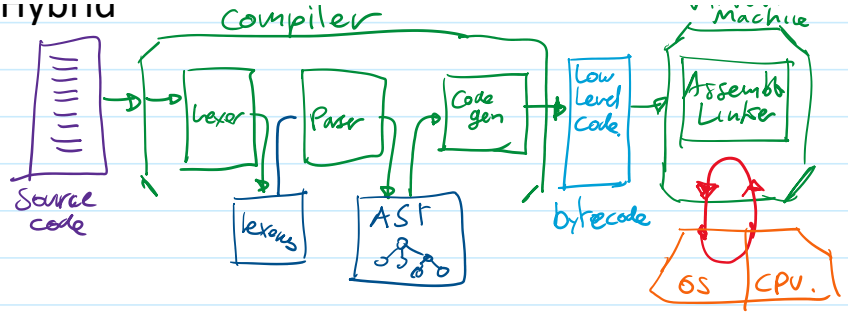
Pro: + Interactivity via REPL
+ Speedy Program Development
+ Portability

Cons: - Speed of execution
- no IP protection. (Code is not hidden)

- Hybrid



- Hybrid



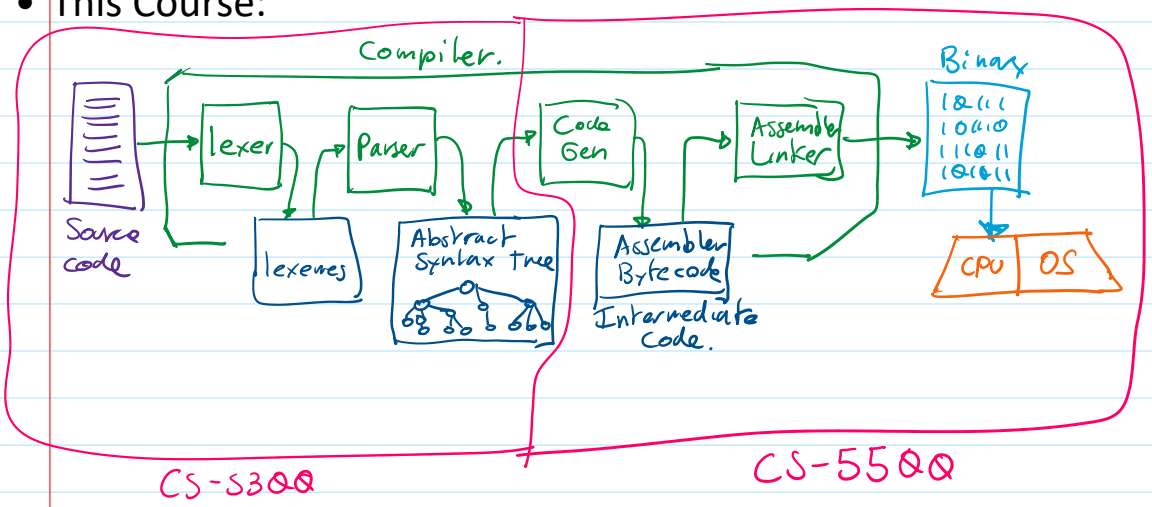
E.G. Java, C#, Scala, Kotlin, Pascal.

Pros: + Speed > Interpreters
+ portability
+ I.P. Protection.
+ inter-Language operability

Cons - Speed < Compilers
- Memory use of Virtual Machines
- no R.E.P.L.

- The "Web Browser"
  - Most web browsers are interpreters JavaScript
  - Turn web browsers into virtual Machines Web ASM.

- This Course:



- EOF