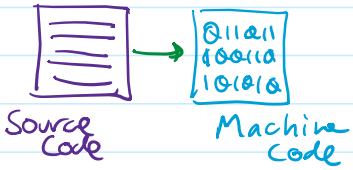
2 Programming Language Implementation

Monday, August 31, 2026 12:26 PM

¢ How are programming Languages implemented?

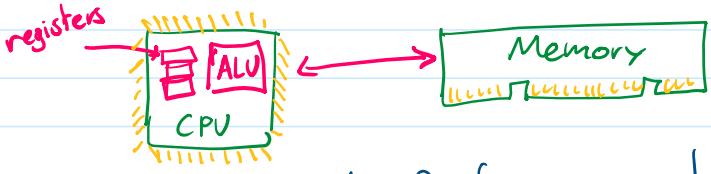
Objective



3 basic types
- Compilers
- interpreters
- hybrids (combination of both)

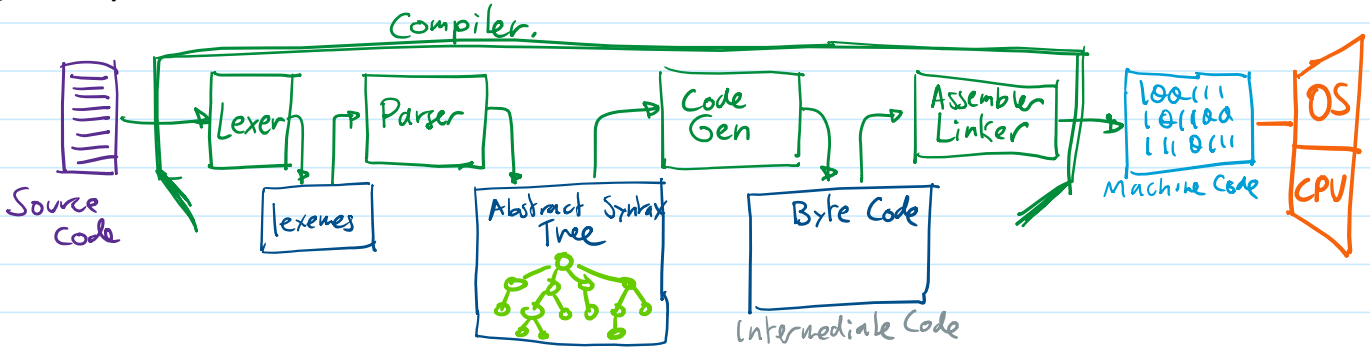
- Regarding Machine Code:

Machine Code is Specific CPU + OS
Machine Code is "relatively" Simple



1.- Perform operations in the ALU + − * =
2.- Fetch/Store data from Memory to Registers
3.- Manipulate the "instruction pointer" (the next instruction to be executed)

1) Compilers



- Lexer: split text into atomic components

E.g. apple += banana * (orange + 7.35);

- Parser: recognize whether Lexemes for a valid program under the rules of the language.

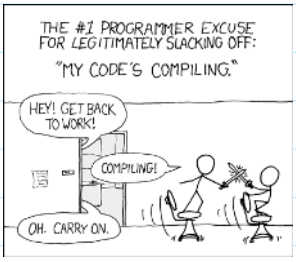
apple = d3fx [* * orange * + ( ] 7 } ;)

E.g. Compilers

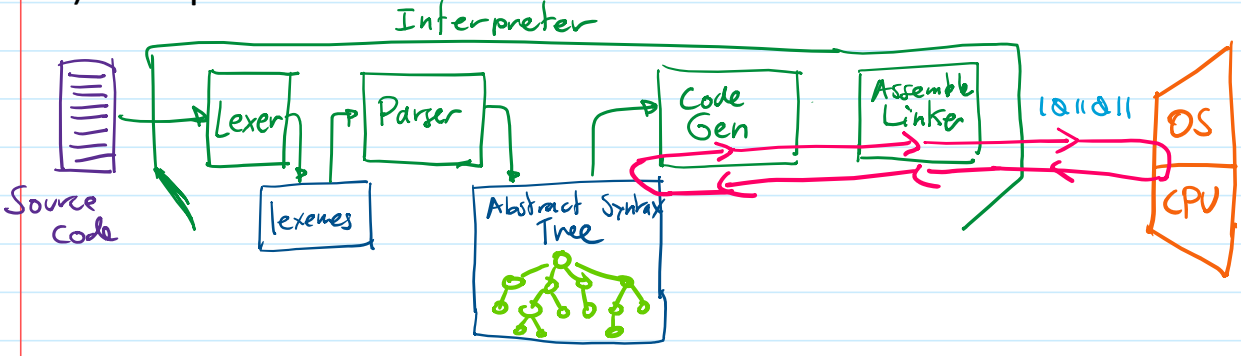
FORTRAN, C, C++, Pascal, D, Rust, Go, zig...

Pro ⊕ Speed of final program.
⊕ ip protection*

Cons ⊖ Portability
⊖ Development Flexibility
⊖ Compilation can be slow.



2) Interpreters



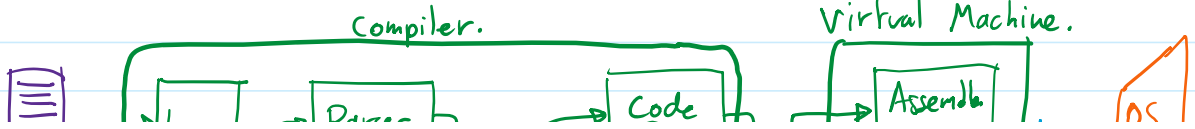
- REPL :- Read - Evaluate - Print - Loop

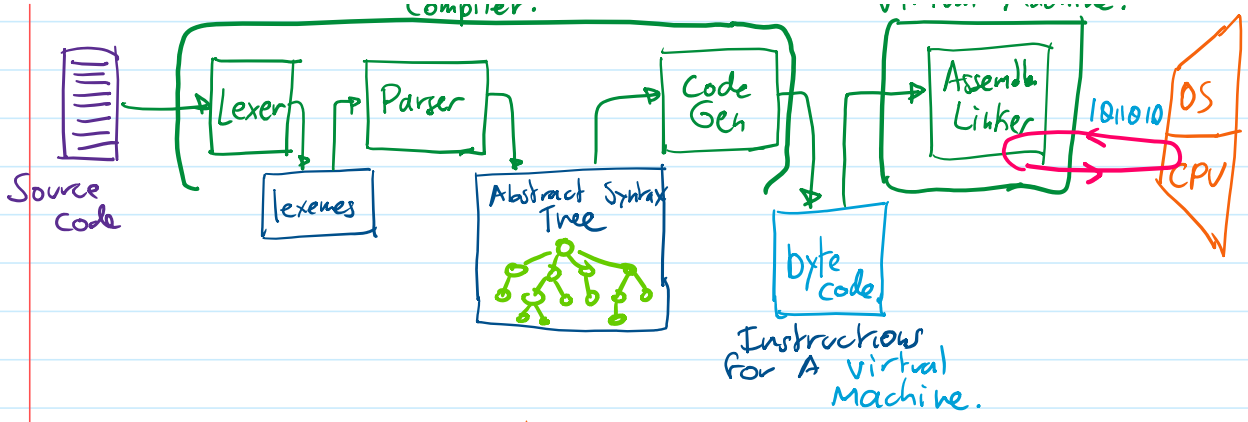
Eg. LISP, Python, JavaScript, Lua, Ruby, ML, BASIC

Pro ⊕ Portability
⊕ Speedy program Development.
⊕ interactivity via REPL

Cons ⊖ Execution speed.
⊖ no IP protection (source code is not hidden)

2) Hybrid





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

Pro ⊕ Speed > Interpreters.
⊕ Portability
⊕ I.P. Protection
⊕ inter-language operability

Cons ⊖ Speed < compilers.
⊖ Memory use of virtual Machines
⊖ No REPL

- The "Web Browser"
  - A JavaScript interpreter.
  - A virtual Machine. WebASM

Node
V8
Electron.

- This course: