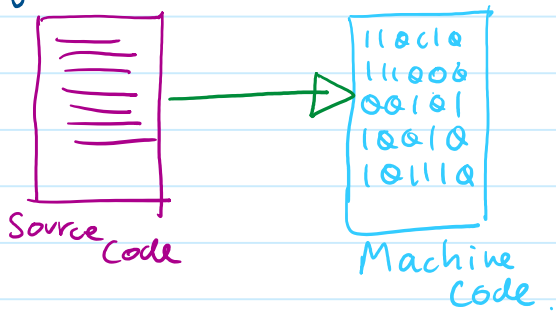
2 Programming Language Implementation

Wednesday, January 29, 2025 12:31 PM

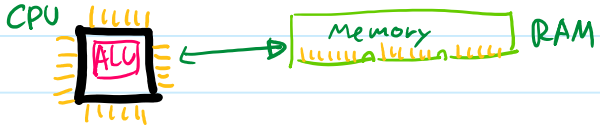
¿ How are programming Languages implemented?

Objective:



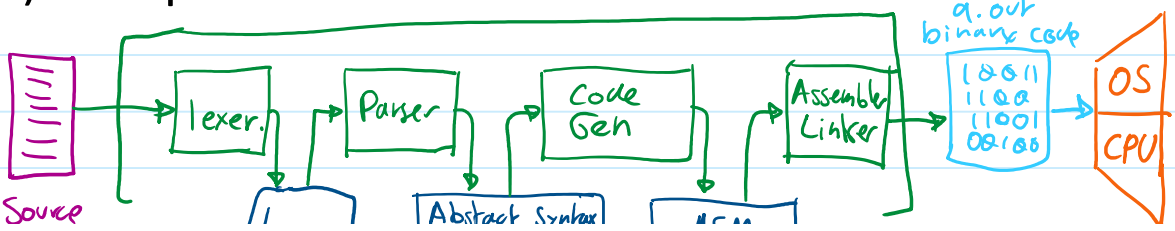
3 Basic types
- Compilers
- Interpreters
- Hybrid (combination of Both)

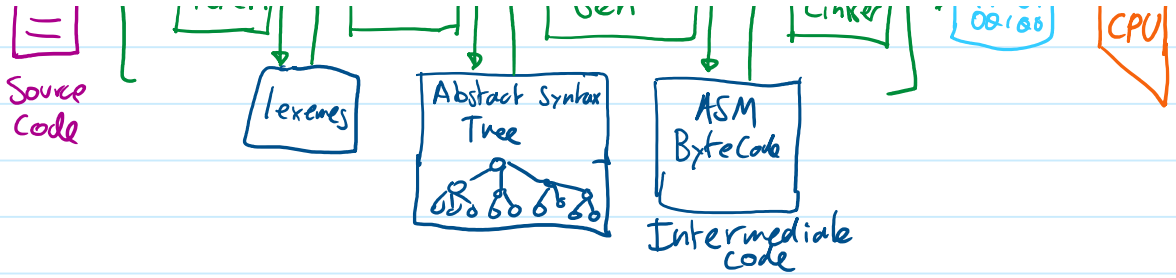
Note:
- Machine code is specific: CPU + OS
- Machine code is "relatively simple"



1.- perform operations on values in the cpu + − ÷ * is_zero? = < >
2.- fetch data from memory to CPU
3.- store data from CPU to Memory
4.- Manipulate the "instruction pointer" (conditional and loops)

1) Compilers





lexer:- split text into atomic components

E.g. apple += banana * (orange + 7.3);

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

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

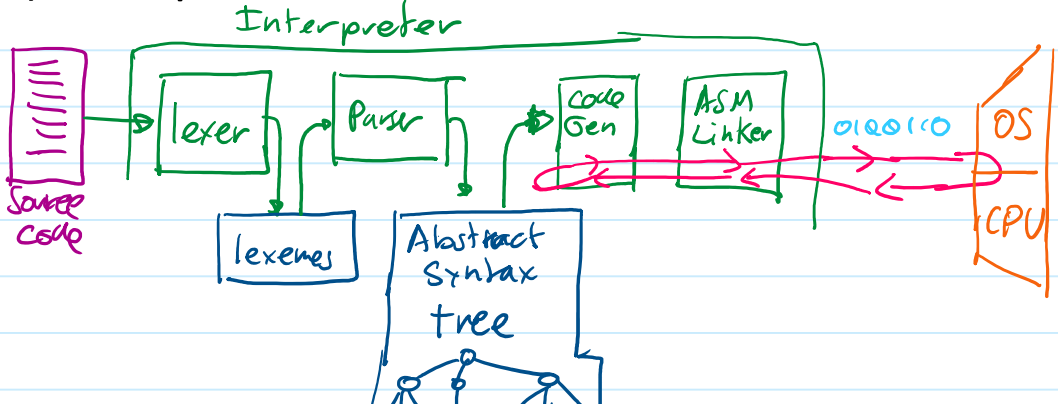
e.g Compilers

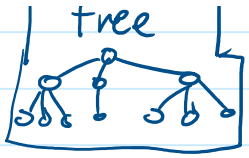
FORTRAN, C, C++, Pacal, D, Rust

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

Cons: - Portability
- Development flexibility
- need a complete program.
- compilation speed can be slow.

2) Interpreters





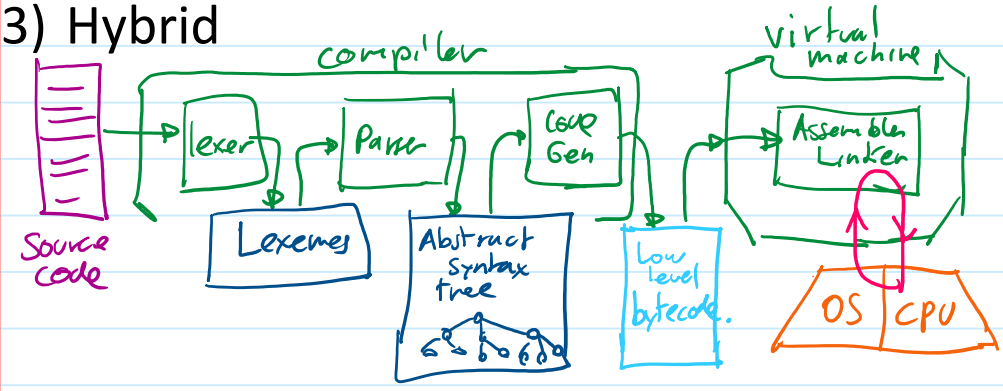
REPL:- Read-Evaluate-Print Loop.

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

Pro: + interactivity via REPL
+ speedy program development
+ portability.

Cons: - Execution speed
- no IP protection (code is not hidden)

3) Hybrid



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

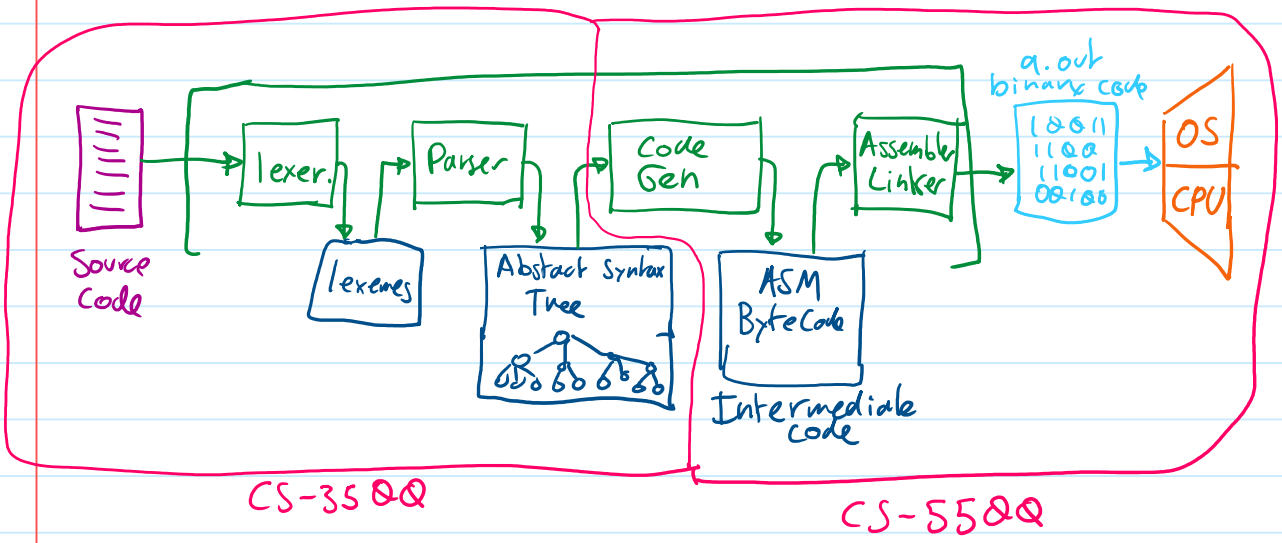
Pros + Speed > Interpreters.
+ Portability
+ I.P. Protection.
+ inter-language operability

Cons - Speed < Compilers.
- Memory use of virtual Machine.
- no R.E.P.L.

- The "Web Browser"

- Web Browsers are JavaScript interpreters.
- Turn Web Browsers into virtual Machines. WebASM.

- This Course



— EOF