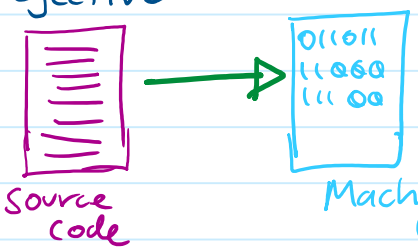
2 Programming Language Implementation

Wednesday, September 3, 2025 11:59 AM

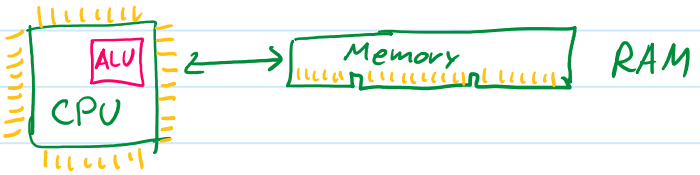
How are programming languages implemented?

Objective:



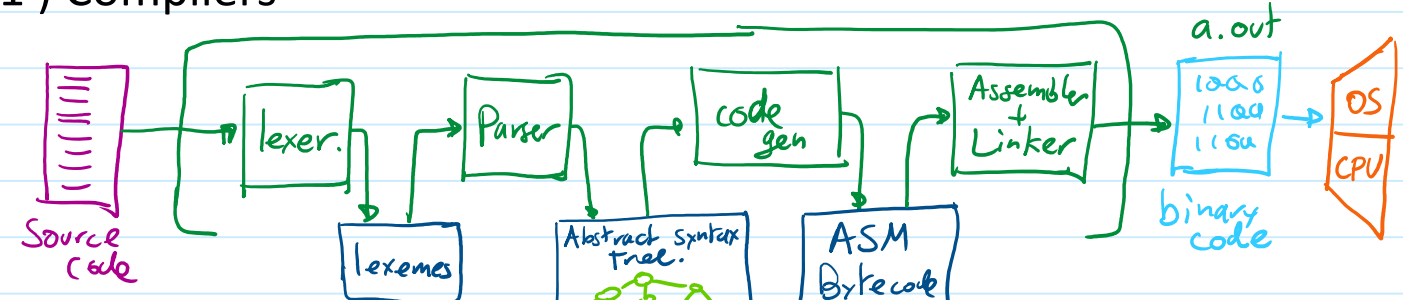
3 Basic types
- Compilers
- Interpreters
- Hybid (Combination of Both)

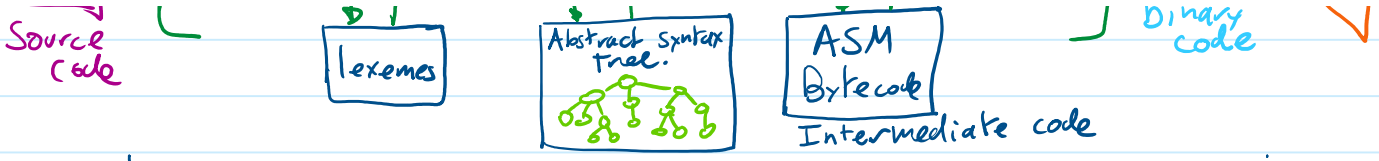
Note:
- Machine Code is specific: CPU + OS
- Machine Code is "Relatively Simple"



1:- Perform operations in the ALU
+ * - % > < ==
2:- Fetch data from memory
3:- Store data to memory
4:- Manipulate the "instruction pointer."
(conditionals / loops)

1) Compilers





lexer: split text into atomic components.

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

a/. data

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

E.g. `apple = 3dfx [ ** orange * + ( ) 7 } ;`

E.g, Compilers

ForTran, C, C++, Pascal, D, Rust, Zig.

Pro: ⊕ speed of final program.

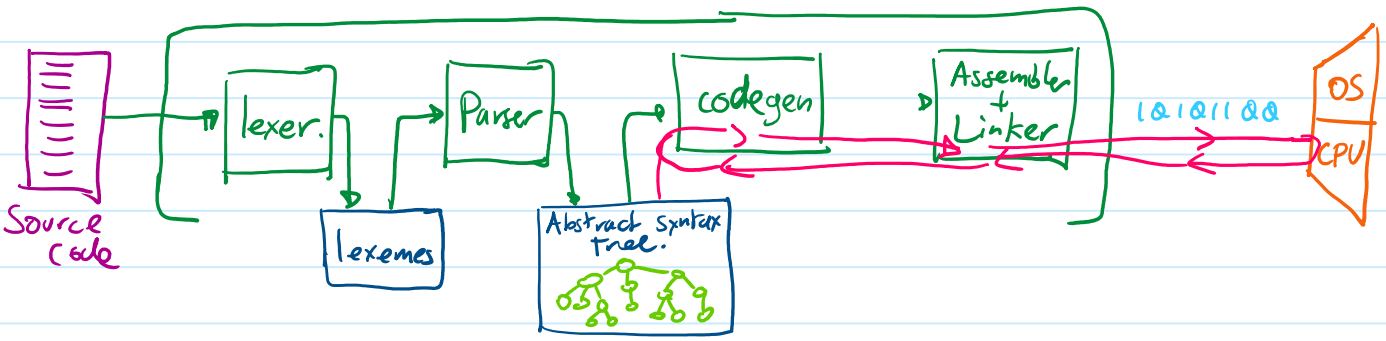
⊕ Some IP protection.

Cons: ⊖ Portability.

⊖ Development Flexibility

- compilation can be slow.

2) Interpreters



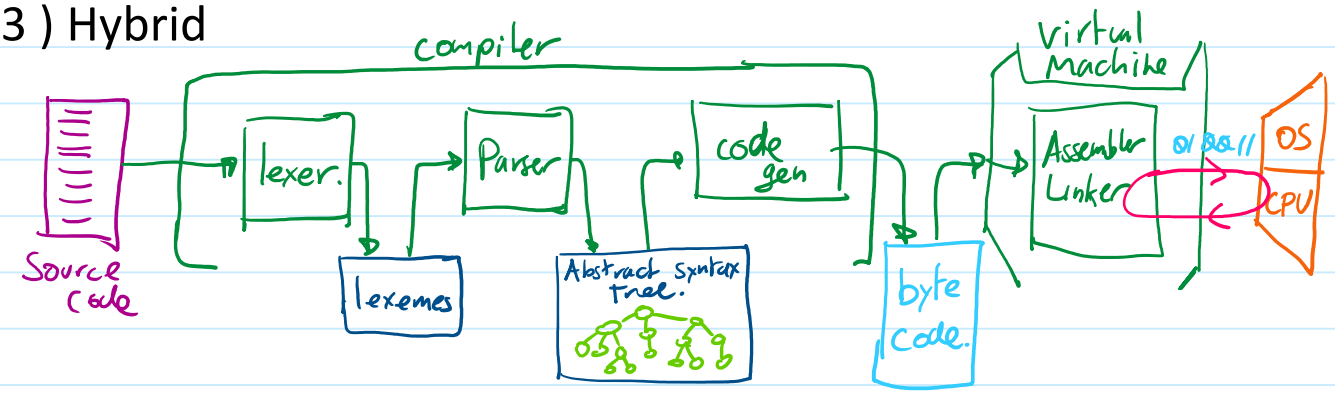
REPL: Read-Evaluate-Print-Loop.

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

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

Cons: ⊖ no IP protection (Source code is not hidden)
⊖ Execution Speed

3) Hybrid



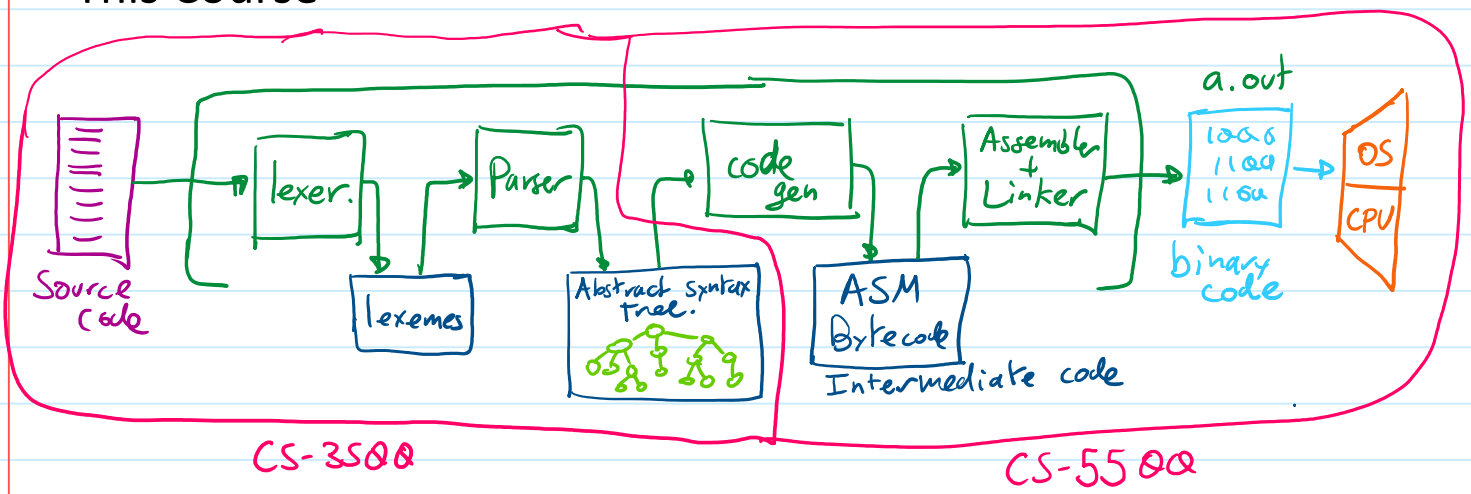
E.g. Java, Scala, Kotlin, C#, Pascal

Pros ⊕ Speed > Interpreters
⊕ Portability
⊕ I.P. Protection
⊕ inter-Language operability

Cons ⊖ Speed < Compiles
⊖ Memory use of Virtual Machine.
⊖ no REPL.

- The "Web Browser"
  - Web browser are JavaScript interpreters
  - Turn Web Browsers into virtual Machines, WebASM.

- This Course



EOF