2024-04-27 Free Range Programming

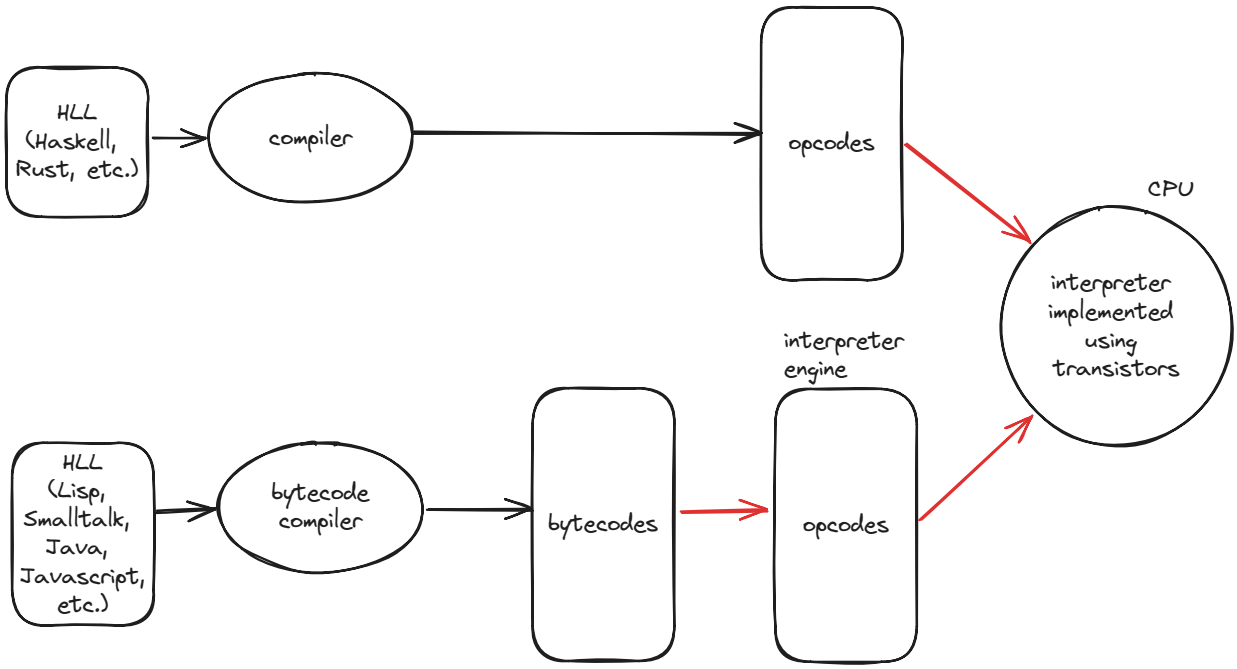
Bloatware

Coding is the act of creating assembler programs.

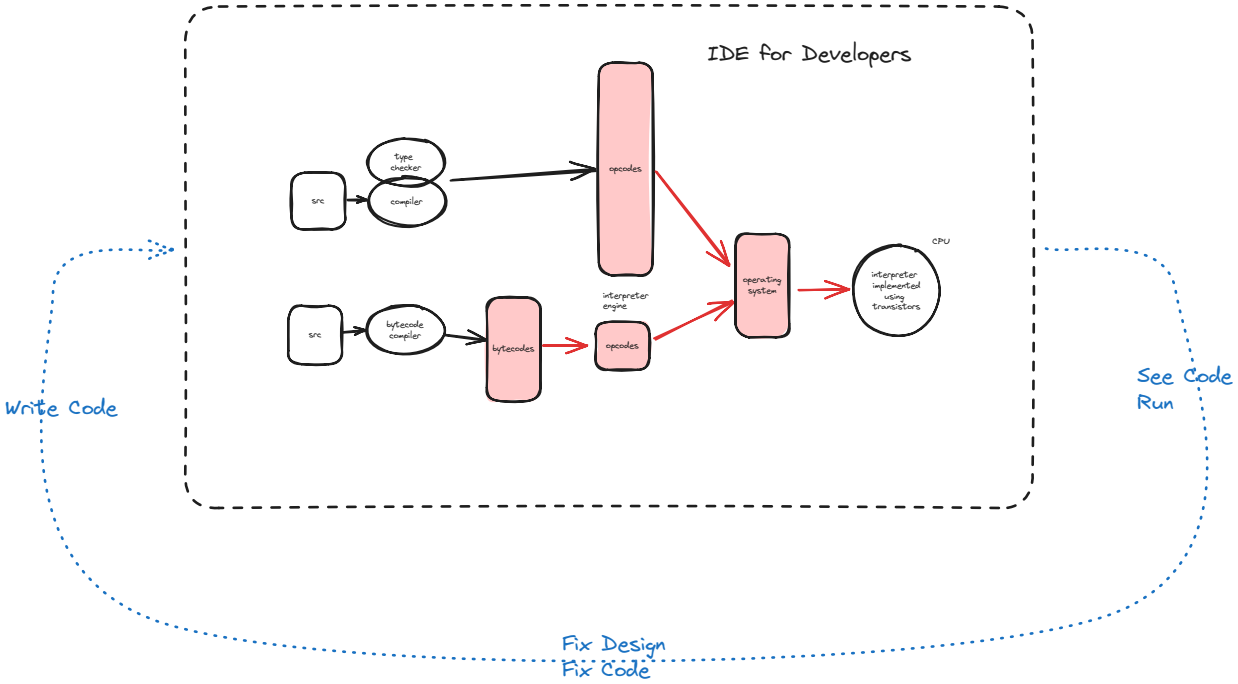
All programming languages, including languages like Haskell and Rust, ultimately create assembler.

The so-called interpreted languages create scripts that are run by engines that are coded in assembler. This adds an extra layer of indirection - the CPU interpreter interprets the engine which interprets encoded instructions.

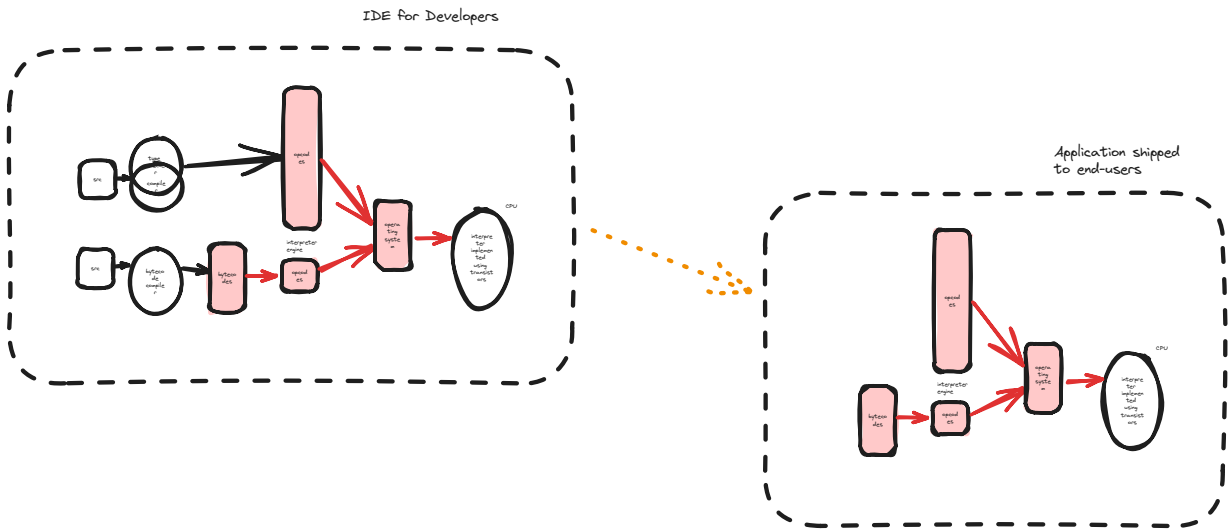
All languages are interpreted. So-called “compiled” languages are interpreted directly by the hardware CPU. The CPU is an interpreter. It interprets encoded instructions (“opcodes”) in hardware.



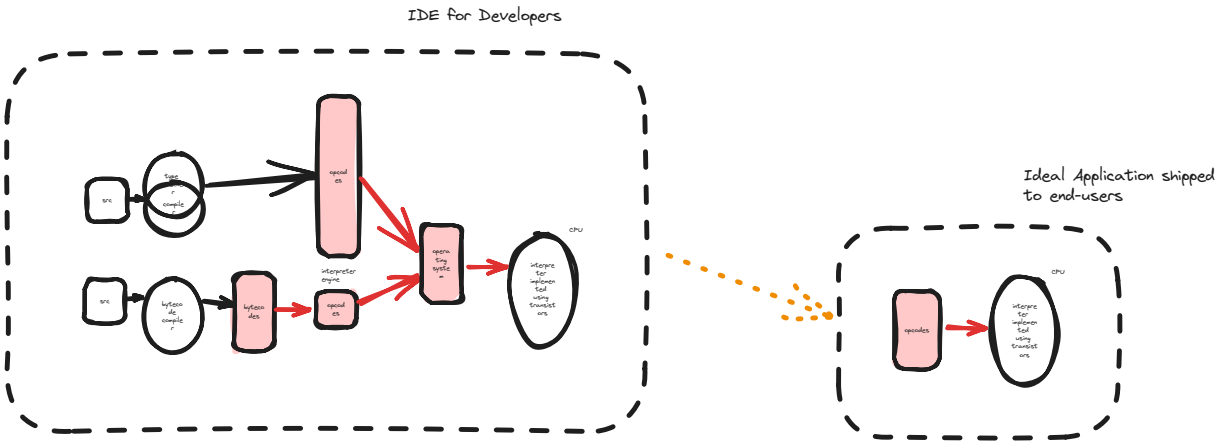
Type-checking creates extra code that developers find to be useful, especially on their souped-up development machines.



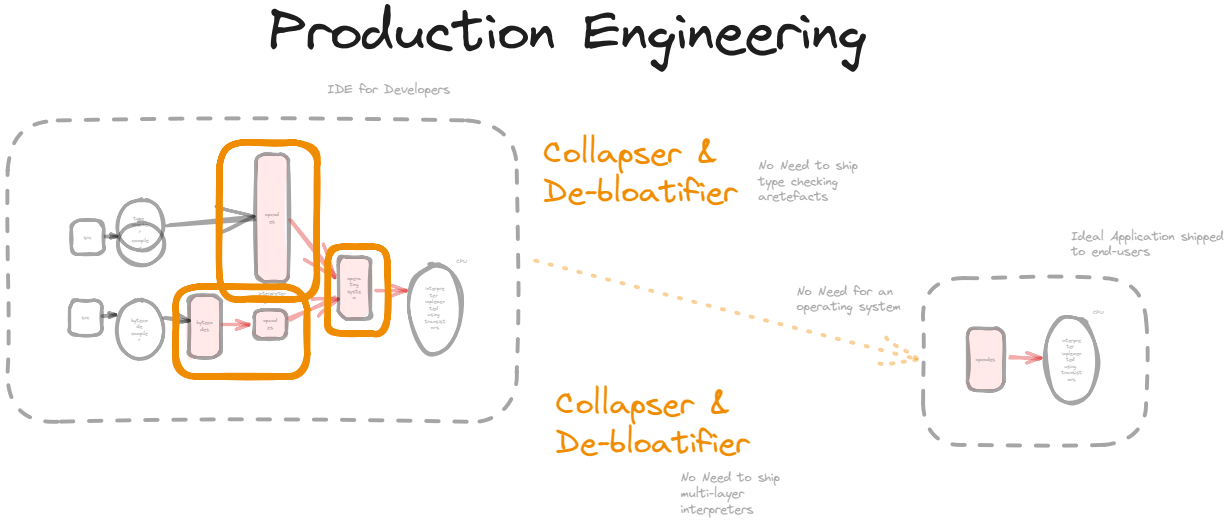
Then, developers ship code that still contains the useful stuff.



But, end-users don't need all of that extra code. End-users don't need to pay for souped-up machinery.



What's missing from this process?



Appendix - See Also

See Also

References *https://guitarvydas.github.io/2024/01/06/References.html*
Blog *https://guitarvydas.github.io/*
Blog *https://publish.obsidian.md/programmingsimplicity*
Videos *https://www.youtube.com/@programmingsimplicity2980*
[see playlist "programming simplicity"]
Discord *https://discord.gg/Jjx62ypR (Everyone welcome to join)*
X *(Twitter) @paul_tarvydas*
More writing *(WIP): https://leanpub.com/u/paul-tarvydas*