CSE 110A - Fundamentals Of Compiler Design

Elijah Hantman

Prof

> Professor: Marcelo Siero
> Course Author: Tyler Sorensen

Syllabus

High Level On Compilers

- Resources
  - Public website for slides, syllabus, and additional resources
  - Private Website grades, announcements, SETs, homework, tests, zoom (if needed)
  - Docker Image for Homework
  - Piazza for questions and discussions
- Quizzes

> Happens out of class, reviewed the next quiz Quizzes are more for discussion than getting the exact correct answer every single time.

- TA
  - Rithik Sharma, Yanwen Xu
  - Mentors/Graders
- Homework
  - Docker Image for standard environment
  - Command line Editor, emacs, vim, nvim.
  - No VSCode support
- Syllabus
  - Autograder given, should always know grade by the deadline

High Level Discussion On Compilers

- What is a compiler?

> Any program that takes some source code and produces machine code.
>
> A transpiler is usually the term for converting between two languages which are not machine code, and an interpreter is when we convert code into actions, executing them on the spot.

- Interpreters vs Compilers

> Interpreters compile and then run the code.

Heart Of a Compiler

- Input → Frontend → Optimizer → Code Gen → Output
- More Detail
  - Input
  - Lexical Analysis (Lexer) (Tokenizing)

  - Syntactic Analyzer (AST)
  - Semantic Analyzer (Correctness testing)
  - Intermediate Code Gen (IR)
  - IR Optimizations
  - Target Code Gen
  - Target Code Optimizations (SIMD) (Platform/CPU specific)
- LLVM Modular Infrastructure
  - Front ends (language specific)
  - LLVM IR (Language and Implementation Agnostic)
  - LLVM Code Gen (Platform and Hardware Specific)
- Front End
  - Lexical Analysis → Token Stream

    Break the input string into units of meaning, names, operators etc.
  - Syntactic Analysis → Syntax Tree

    Combine the tokens into a structure which represents their meaning. Usually a tree is used since it is well known that trees can represent operators on inputs
  - Semantic Analysis → Correct Syntax Tree

    The Semantic Analysis stage ensures that the syntax tree follows all the rules of the language. Generally if this stage is passed the Compiler garuntees a full compilation without errors.
  - Intermediate Code Gen → IR program

    The tree is iterated over, usually in a post traversal, and converted into some intermediate representation which contains instructions.
    We then optimize this IR which is usually easier due to certain promises and structure the IR garuntees