EECS 483: Compiler Construction

Lecture 0: Course Overview, Baby's First Compiler

January 8, 2025

Introductions

Instructor: Max S. New

GSI: Yuchen Jiang

IA: David Mekhtiev

Programming Language Implementation

When you write a program in your favorite programming language, how does it get executed?

Interpreted

Another program, an "interpreter" reads in your program and implements its behavior

Compiled

Another program, a "compiler" and outputs another program, which we already know how to run. Most commonly, the binary format that your CPU executes.

Just-in-time compilation (Javascript v8): an interpreter that selectively compiles part of the program to speed up execution

Bytecode interpreters (Java): compile to a language for which you have a fast interpreter implemented.

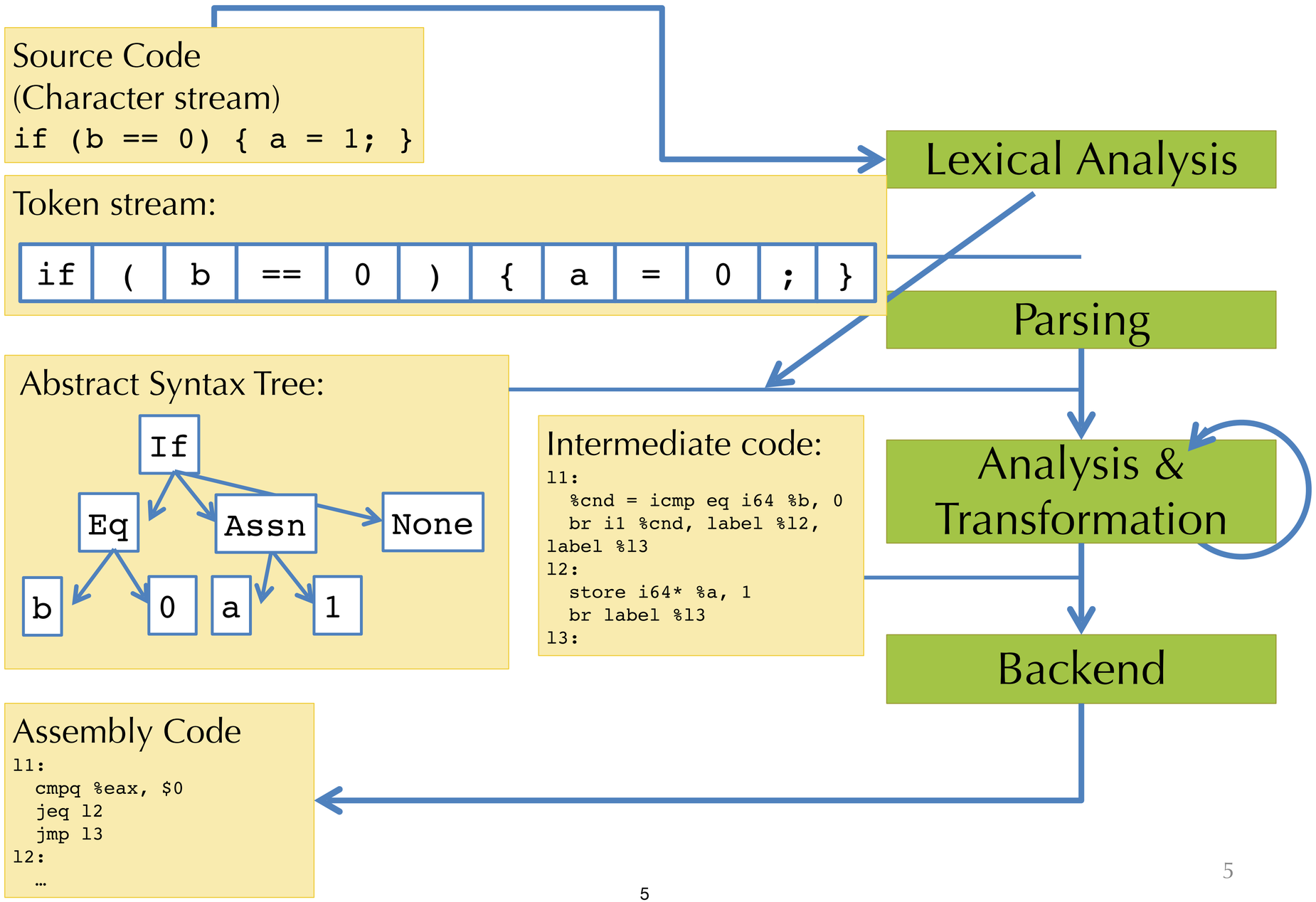
Fundamentally, a compiler is a kind of **translator**

```
compiler : SourceProgram -> TargetProgram
```

Usually the source programs are easy to write, and the target programs are easy to use.

```
gcc, clang   : C/C++      -> Binary          /* a.out, .exe */
emcc         : C/C++      -> WebAssembly
rustc        : Rust       -> Binary
javac        : Java       -> JvmByteCode     /* .class */
scalac       : Scala      -> JvmByteCode
gwt          : Java       -> JavaScript      /* .js */
v8           : JavaScript -> Binary
nasm         : X64        -> Binary
pdftex       : LaTeX      -> PDF
pandoc       : Markdown   -> PDF or Html or Doc
```

What do compilers look like on the inside?



5

Learning Objectives

- How high-level programming constructs are compiled to assembly code
- Modern compiler intermediate representation (Static Single Assignment "SSA")
- How to write, refactor, test and debug a large compilation pipeline
- How to represent and manipulate programs as data
- How program analyses are implemented and how they are used to optimize programs
- How to parse complex data formats into convenient representations

Course Webpage

https://maxsnew.com/teaching/eecs-483-wn25

Coursework

You will be responsible for 5 programming assignments and 2 exams (midterm and final).

- Programming Assignments: 70% of final grade, each 14%
- Exams: 30% of final grade, each 15%

Homework Assignments

In our 5 homework assignments, we will build up to an optimizing compiler for the "Snake" language.

For assignments 1-4 we will gradually add programming language features, each assignment's solution will build on the previous.

For assignment 5 we will improve our generated code by performing optimizations.

Code will be implemented in Rust. Can be completed solo or in groups of 2. Do not collaborate with anyone unless you are in a group together.

Exams

Midterm exam: programming language basics, scope, SSA form, x86 assembly

Final exam: second half of the course. Data representations, optimization, lexing/parsing/typechecking

Exams are complementary to programming assignments. Programming assignments demonstrate practical knowledge while exams demonstrate theoretical understanding.

Tools

Assignments are implemented in Rust

Target language of our compiler is x86_64 assembly code

You are not expected to have used either of these previously.

1. Use this first weeks of class and the first assignment to familiarize yourself with Rust

2. We will learn details of x86_64 assembly throughout the semester.

Baby's First Compiler

When we implement a compiler (to assembly) we need to address the following questions:

1. What is the syntax of the language we are compiling?
2. What is the semantics of the language we are compiling?
3. How can we implement that semantics in assembly code?
4. How can we generate that assembly code programmatically?

Snake v0: "Neonate"

1. What is the syntax of the language we are compiling?

   Integer literals

2. What is the semantics of the language we are compiling?

   Print out the number

3. How can we implement that semantics in assembly code?

   Produce a function that outputs that number, call that function from Rust.

4. How can we generate that assembly code programmatically?

For Next Time

- Read the course webpage: http://maxsnew.com/teaching/eecs-483-wn25/
- Work through the first 4 chapters of the Rust book: https://rust-book.cs.brown.edu/
- Attend discussion section on Friday for more intro to Rust