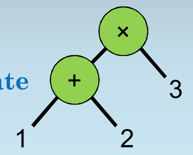
Front End and Back End Operate Independently

Instead,
- **front end** converts language (such as C) to an **intermediate representation (IR)**, such as … trees!
- (IR can be optimized.)
- **back end** converts IR to assembly code.*



(10 + 10) / 2 = 10 compilers to write (<< 100)!

*Take CS426 (421 for front-end, with other stuff).

A Modern Example

Chris Lattner (UIUC CS Ph.D., 2005)
- developed LLVM compiler framework
- with Vikram Adve's group as a grad student,
- and continued to work on it within Apple.

In 2010, he
- started to develop the Swift programming language,
- using the LLVM compiler (IR and back end) as a starting point.

One Benefit of High-Level Languages: Managing Variables

What good are high-level languages?

Remember deciding (in examples and MPs)
- what information to store, and
- where to put it
- (which register, or which memory location)?

In high-level languages,
- **programmer specifies symbolic name** (like a label in assembly) **and**
- **data type**.

Compiler decides where to put each variable.

High-Level Languages Support Complex Data Types

The benefit generalizes to include…
- **structures** (such as events in MP2), and
- **arrays** (event list in MP2), and
- **pointers** (in the schedule in MP2).*

Compiler
- **knows how each maps into memory**,
- and **manages access for you** by name.

*We'll see how later in our class.