How Do We Compile The Compiler?

- This sounds much like:
  “Which came first, the chicken or the egg?”

How Do We Compile The Compiler?

- This sounds much like:
  “Which came first, the chicken or the egg?”
- If we had written our LiveOak compiler manually in x86 assembly language, we wouldn’t have a problem. But ...

How Do We Compile The Compiler?

- This sounds much like:

  “Which came first, the chicken or the egg?”
- If we had written our LiveOak compiler manually in x86 assembly language, we wouldn’t have a problem. But ...
  - The LiveOak compiler is a Java program.
  - Java programs are run in the JVM, which is basically a large piece of C code.
  - C code is compiled to x86 machine code by the C compiler.
  - The C compiler is itself a C program.

How Do We Compile The Compiler?

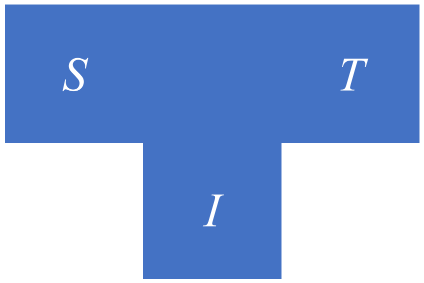
- This sounds much like:

  "Which came first, the chicken or the egg?"
- If we had written our LiveOak compiler manually in x86 assembly language, we wouldn't have a problem. But ...
  - The LiveOak compiler is a Java program.
  - Java programs are run in the JVM, which is basically a large piece of C code.
  - C code is compiled to x86 machine code by the C compiler.
  - The C compiler is itself a C program.
- Wait! Aren't we going into an infinite loop here?
- No: we will stop the process at a well-defined point and bootstrap our way up from there.

What Are Compilers, Anyway?

- They are just programs, but they are programs of a very special kind.
  - They take as input (strings in) a source language S.
  - They are written in an implementation language I.
  - They produce as output (strings in) a target language T.

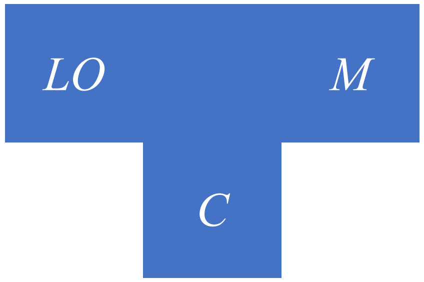
- Let's show this graphically as a T-diagram.
- Or, symbolically as $\mathcal{C}(S, I, T)$.



- We will use M to denote machine language.
  - The only program that can run on a hardware machine is one whose implementation language is M.

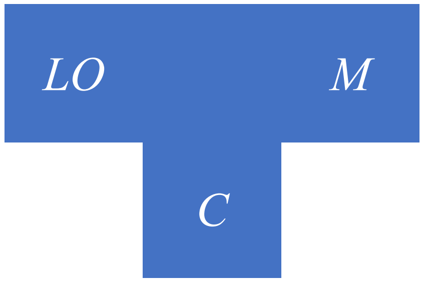
A Simpler Problem First

- Suppose we had written our LiveOak compiler in C rather than in Java. Then its representation is

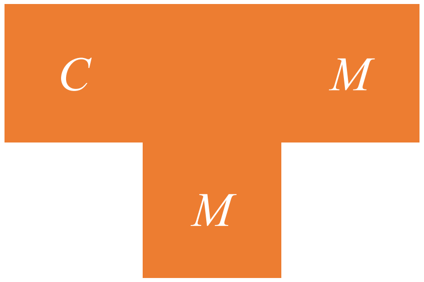


A Simpler Problem First

- Suppose we had written our LiveOak compiler in C rather than in Java. Then its representation is

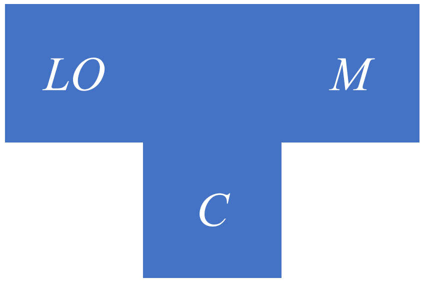


- Since the C compiler is available as a binary, its representation is

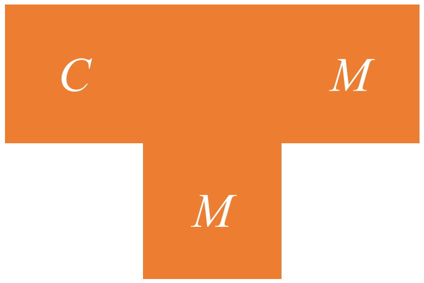


A Simpler Problem First

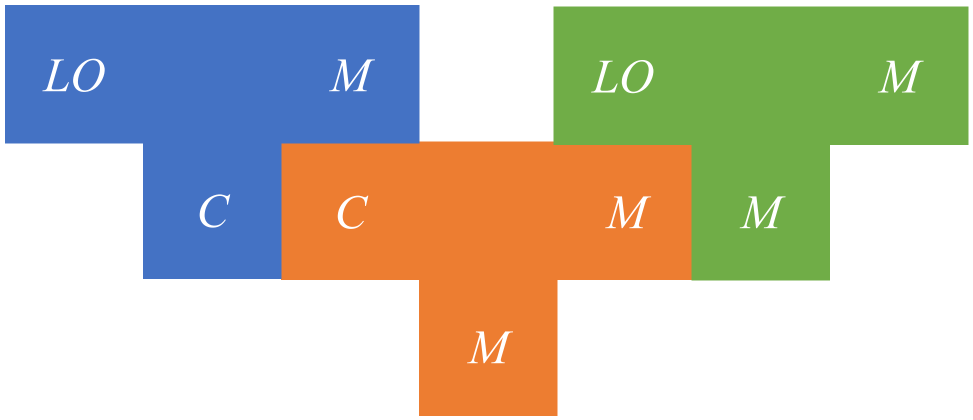
- Suppose we had written our LiveOak compiler in C rather than in Java. Then its representation is



- Since the C compiler is available as a binary, its representation is

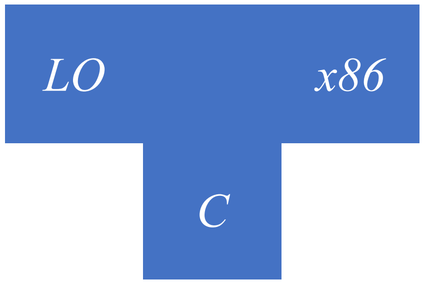


- So we can compose these and get

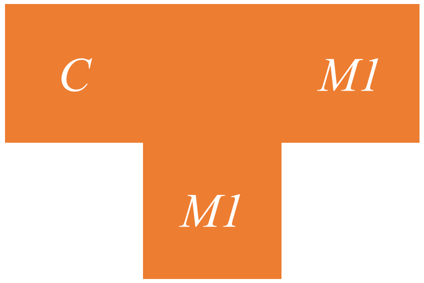


Variation 1: A Cross-Compiler

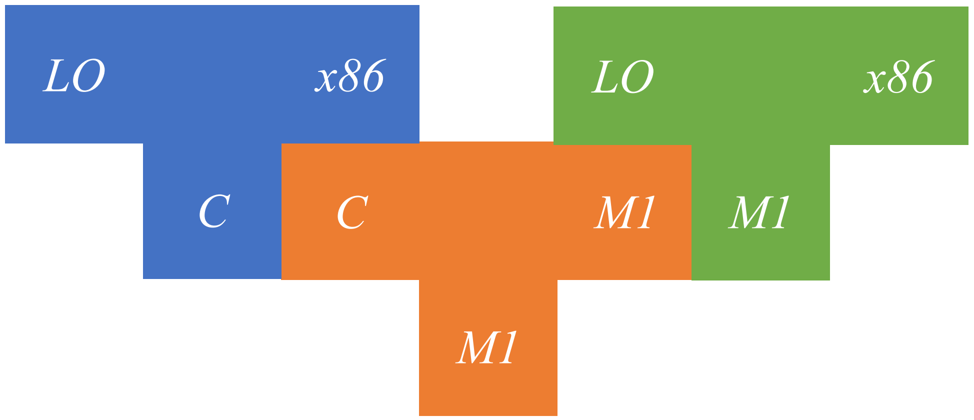
- LiveOak compiler for x86 written in C on an M1-based Mac:



- C compiler (executable) on the M1-based Mac:

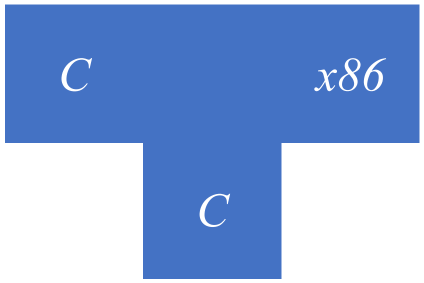


- So we have:

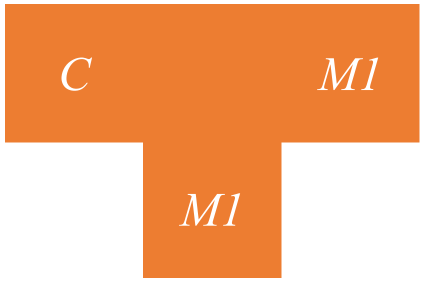


Variation 2: Two-Step Bootstrapping

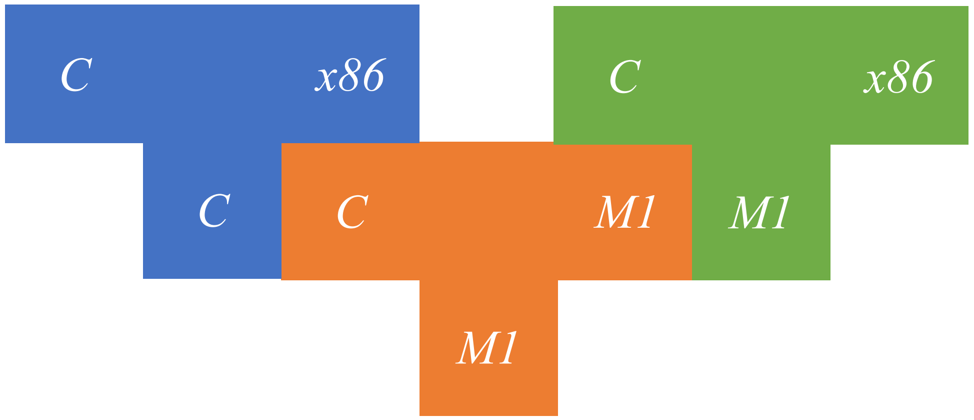
- C compiler for x86 written in C on an M1-based Mac:



- C compiler (executable) on the M1-based Mac:

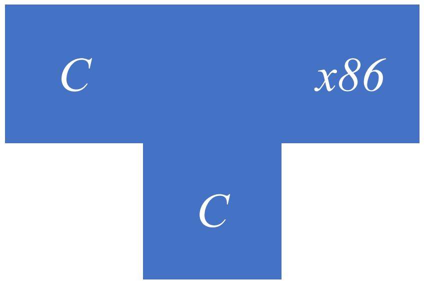


- So we have (step 1):

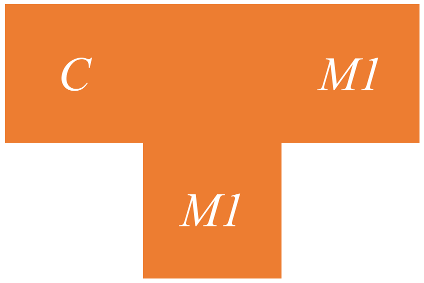


Variation 2: Two-Step Bootstrapping

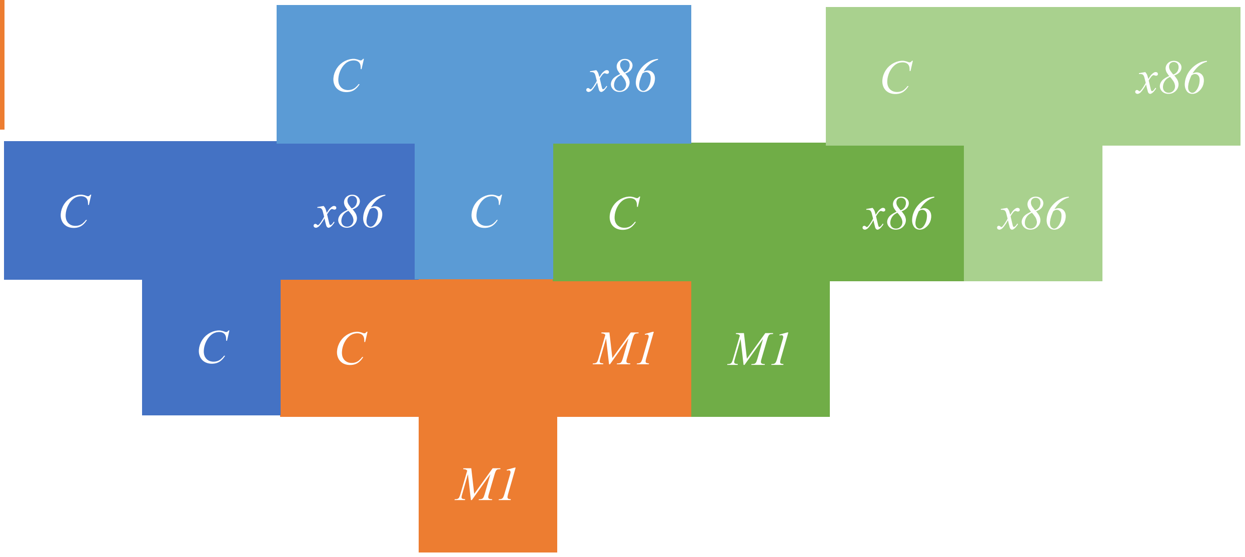
- C compiler for x86 written in C on an M1-based Mac:



- C compiler (executable) on the M1-based Mac:



- And then (step 2):



C
x86
C
x86
C
x86
C
C
x86
x86
C
C
M1
M1
M1