THEORY OF COMPUTATION AND COMPILERS

Unit - I | Part-2

OVERVIEW OF COMPILATION

- **Bootstrapping**

Dr. R. Madana Mohana
Professor, Artificial Intelligence & Data Science | I/c-Head, Artificial Intelligence & Machine Learning
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY
Hyderabad - 500 075, Telangana, INDIA
www.cbit.ac.in

Bootstrapping

- Compiler is a complex program and should not be written in assembly language
- How to write compiler for a language in the same language (first time!)?
- First time this experiment was done for Lisp (List processing)
- Initially, Lisp was used as a notation for writing functions.
- Functions were then hand translated into assembly language and executed
- McCarthy wrote a function `eval[e,a]` in Lisp that took a Lisp expression `e` as an argument
- The function was later hand translated and it became an interpreter for Lisp

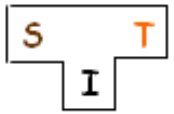
Bootstrapping

- Bootstrapping is the process of writing a compiler for a programming language using the language itself.
- In *other words*, it is the process of using a compiler written in a particular programming language to compile a new version of the compiler written in the same language.
- Bootstrapping is an important technique in compiler design that allows for greater control over the optimization and code generation process, while ensuring compatibility between the compiler and the target language.

Bootstrapping

- A compiler can be characterized by three languages: the source language (S), the target language (T), and the implementation language (I)
- The three language S, I, and T can be quite different. Such a compiler is called cross-compiler

- This is represented by a T-diagram as:



- In textual form this can be represented as

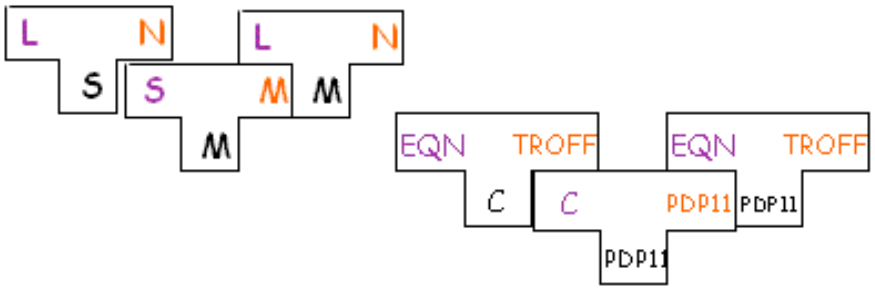
S_I^T

Bootstrapping

- Compilers are of two kinds: *native* and *cross*.
- Native compilers are written in the same language as the target language. For example, SMM is a compiler for the language S that is in a language that runs on machine M and generates output code that runs on machine M.
- Cross compilers are written in different language as the target language. For example, SNM is a compiler for the language S that is in a language that runs on machine N and generates output code that runs on machine M.

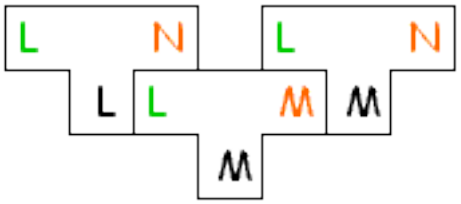
Bootstrapping

- Write a cross compiler for a language L in implementation language S to generate code for machine N
- Existing compiler for S runs on a different machine M and generates code for M
- When Compiler L_SN is run through S_MM we get compiler L_MN

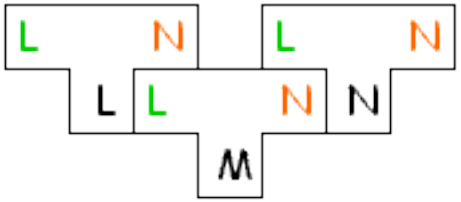


Bootstrapping a Compiler

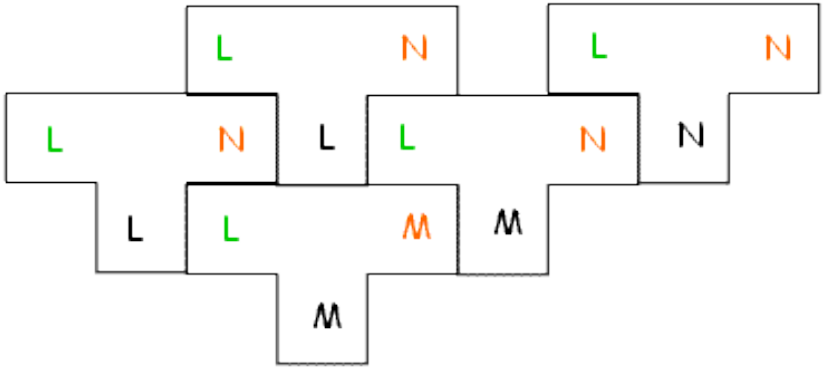
- Suppose $L_L N$ is to be developed on a machine M where $L_M M$ is available



- Compile $L_L N$ second time using the generated compiler



Bootstrapping a Compiler: The complete picture



- Bootstrapping is obtaining a compiler for a language L by writing the compiler code in the same language L.

Bootstrapping

Advantages:

- Bootstrapping ensures that the compiler is compatible with the language it is designed to compile, as it is written in the same language.
- It allows for greater control over the optimization and code generation process.
- It provides a high level of confidence in the correctness of the compiler because it is self-hosted.

Disadvantages:

- It can be a time-consuming process, especially for complex languages or compilers.
- Debugging a bootstrapped compiler can be challenging since any errors or bugs in the compiler will affect the subsequent versions of the compiler.
- Bootstrapping requires that a minimal version of the compiler be written in a different language, which can introduce compatibility issues between the two languages.
- Overall, bootstrapping is a useful technique in compiler design, but it requires careful planning and execution to ensure that the benefits outweigh the drawbacks.