

# Compiler Techniques

## 1. Introduction

Huang Shell Ying

---

The future belongs to those with the  
underlying ability to learn and the  
underlying passion for knowledge.

Warren Fernandez  
Editor, Straits Times.  
15 November, 2015.

# Teaching staff members

▶ Huang Shell Ying ([assyhuang@ntu.edu.sg](mailto:assyhuang@ntu.edu.sg)),

- ▶ Lectures: Week 1 – Week 7
- ▶ Tutorials and Labs



▶ Ta Nguyen Binh Duong ([donta@ntu.edu.sg](mailto:donta@ntu.edu.sg)),

- ▶ Lectures: Week 8 – Week 13
- ▶ Tutorials and Labs



▶ Alwen Tiu ([ATiu@ntu.edu.sg](mailto:ATiu@ntu.edu.sg))

- ▶ Tutorials



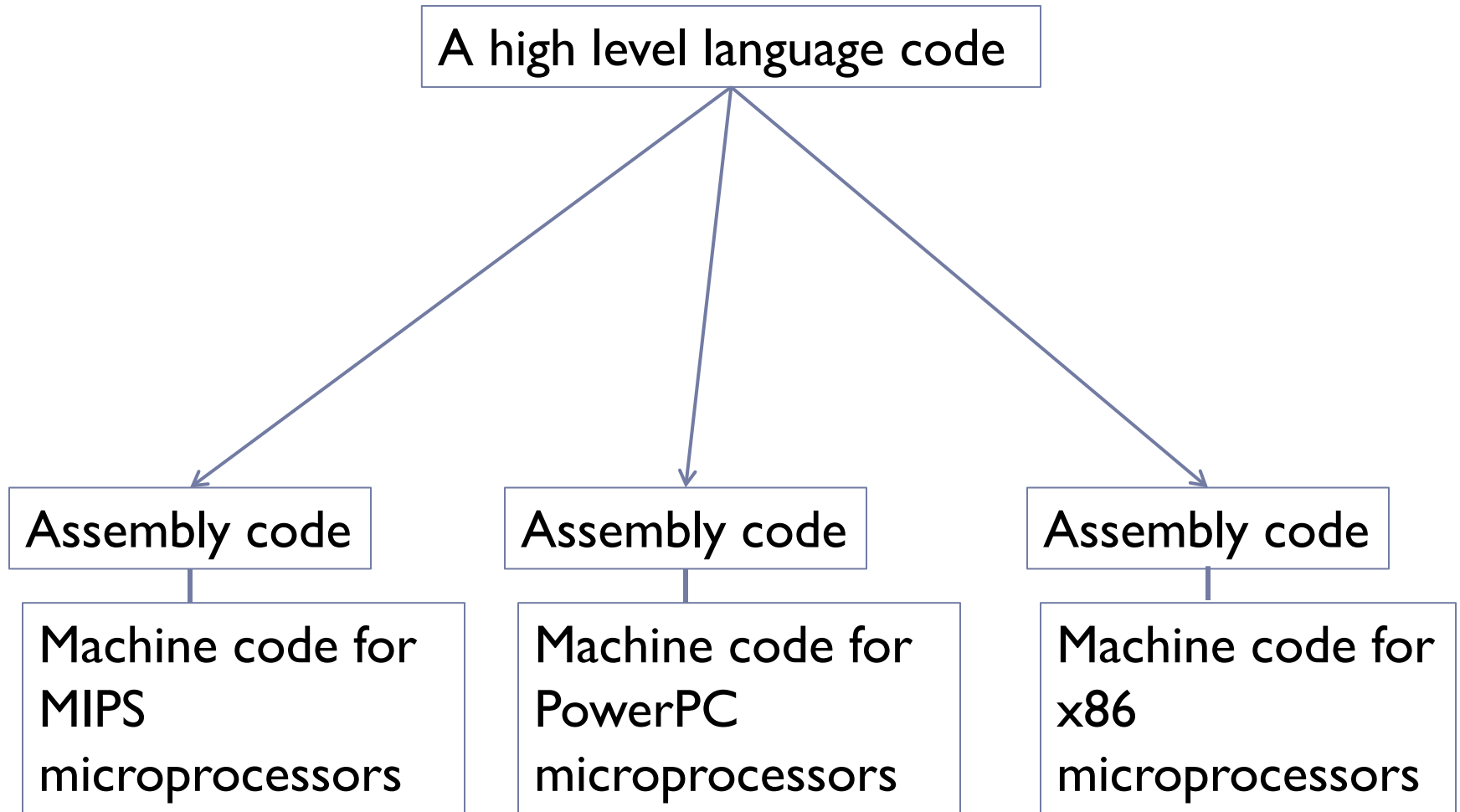
# Programming Languages

---

- ▶ A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer (quote Wikipedia).
- ▶ Machine code or **machine language** is a set of instructions executed directly by a computer's central processing unit (CPU).
  - ▶ Every processor or processor family has its own machine code instruction set.
- ▶ An **assembler language** is a low level programming language, generally with a one-to-one correspondence with the machine code of a processor family.
- ▶ A **high level programming language** is human-readable and needs compiling or interpreting.

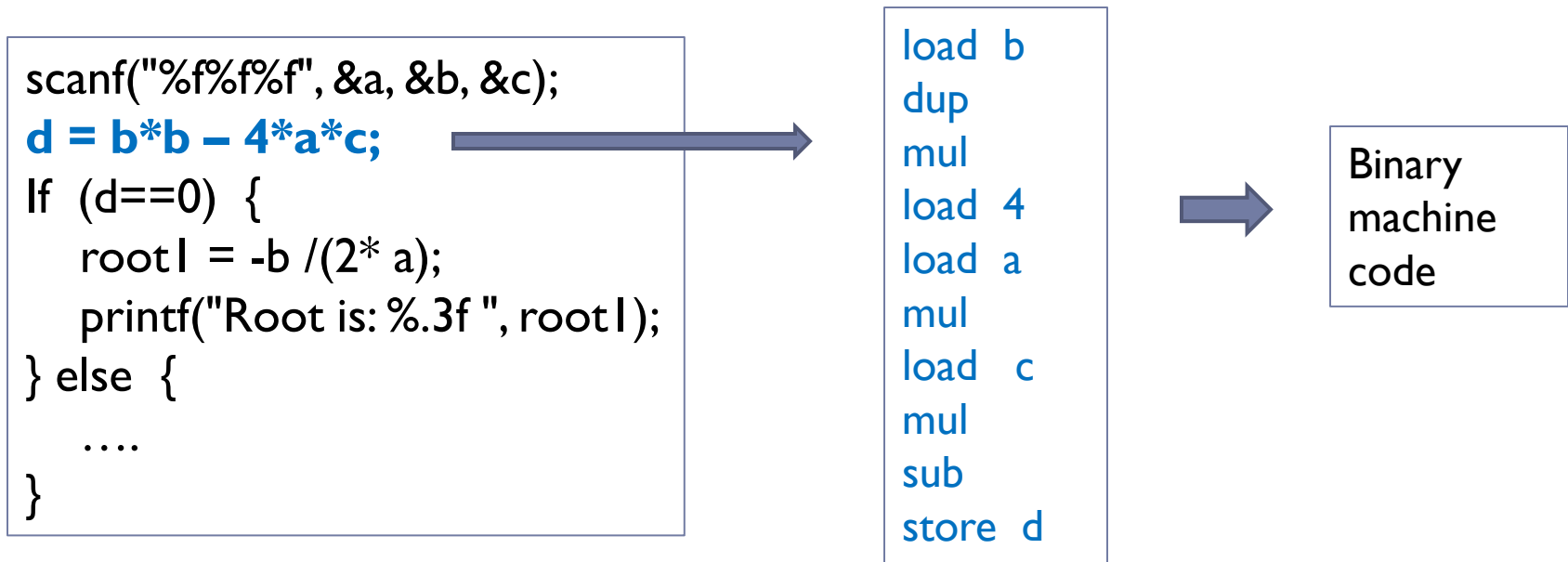
# Programming Languages

---



# A Compiler is a Translator

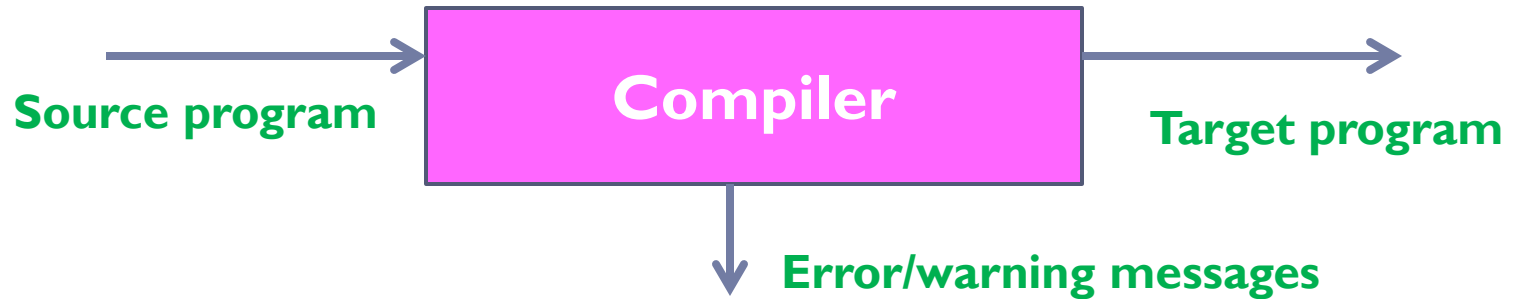
- Programs in a high level programming language may be translated by a compiler ( and an assembler) to machine code programs.



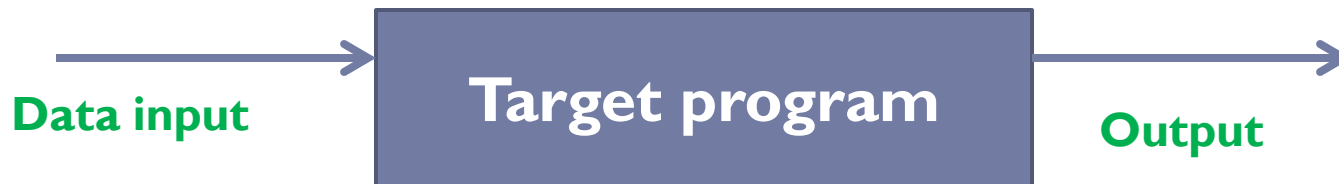
- A **compiler** is a program that reads a program in one language and **translate** it into an equivalent program in a low-level language, such as a microprocessor's or a simple virtual machine's language

# A Compiler is a Translator

---



- ▶ If the target program is in a machine code, the target program will be **executed directly** on the machine.



- ▶ A target program is not necessarily in a machine code. It can be in a virtual machine's language, e.g. Java bytecode.

# Interpreter

- ▶ An **interpreter** directly executes the operations specified in the source program on inputs supplied by the user



- ▶ only applicable if the source program is in a language that has an interpreter on the machine
- ▶ If a program is interpreted, it takes a longer time to run than if it had been compiled
- ▶ Programs may be modified as execution proceeds
- ▶ Interpreters provide a significant degree of machine independence – no machine code is generated, just port the interpreter to a new machine



# Learning Objectives

---

To give students an understanding of compilers and the techniques involved in programming language translation.

- ▶ The major stages of compilation, including lexing, parsing, semantic analysis, optimization and code generation, are described.
- ▶ Relevant tools and techniques introduced.

# Learning Outcomes

---

1. Understand the steps and techniques involved in **programming language translation**.
2. Use **regular expressions** and **context free grammars** to describe languages, and to employ open-source tools to create **recognisers** for them
3. Understand the concept of **abstract syntax trees** and how they are used as the basis for **name binding analysis** and **type checking**.
4. Understand how a compiler generates **machine code**, and how to use simple **data flow analysis** techniques for **optimization**.
5. Apply ideas, techniques, and skills learnt to general software design.

# Why Study Compiler Techniques

---

- ▶ Programming language design and compiler construction are still evolving and are active areas of research and development.
- ▶ To be a good **user** or designer of programming languages, one must know
  - ▶ how a computer carries out the instructions of a program (including how data are represented)
  - ▶ the techniques by which a high-level program is converted into something that runs on an actual computer.
- ▶ Many techniques used in compilers are useful in (i) general software design and software development; (ii) mobile and industry applications.

DesignScript

# Course Schedule

Week	Lecture Topic	Tutorial (hour (topic))	Lab
1	Introduction to compilers & Lexical Analysis		
2	Lexical Analysis		
3	Parsing	1 (Lexical Analysis)	
4	Parsing	0.5(Lexical Analysis)+ 0.5 (Parsing)	lab 1(Lexical Analysis)
5	Parsing & Semantic Analysis	1 (Parsing)	
6	Semantic Analysis	1 (Parsing)	lab 2(Parsing)
7	Semantic Analysis	1 (Semantic Analysis)	
8	Code Generation	1 (Semantic Analysis)	lab 3(Semantic Analysis)
9	Code Generation	1 (Semantic Analysis)	
10	Code Generation	1 (Code Generation)	Quiz (90 minutes)
11	Code Generation	1 (Code Generation)	
12	Optimization	1 (Code Generation)	lab 4(Code Generation)
13	Optimization	1 (Optimization)	

# Grading Information



Attendance  
is important

- ▶ 3 students working in one team for lab assignments
- ▶ Same grade for everyone in the same team, in general
- ▶ 3 lab submissions - 21% of the final grade
- ▶ Attendance in Lab session 3 - 3% of the final grade
- ▶ One quiz in Week 10/11 (compulsory) - 16% of the final grade
- ▶ Final exam – 60% of the final grade
- ▶ If one attends tutorials regularly and asks questions such that your tutor/lecturer remembers you – marks added to the quiz results to improve grades (up to B-)

# General Information

---

Slide 24

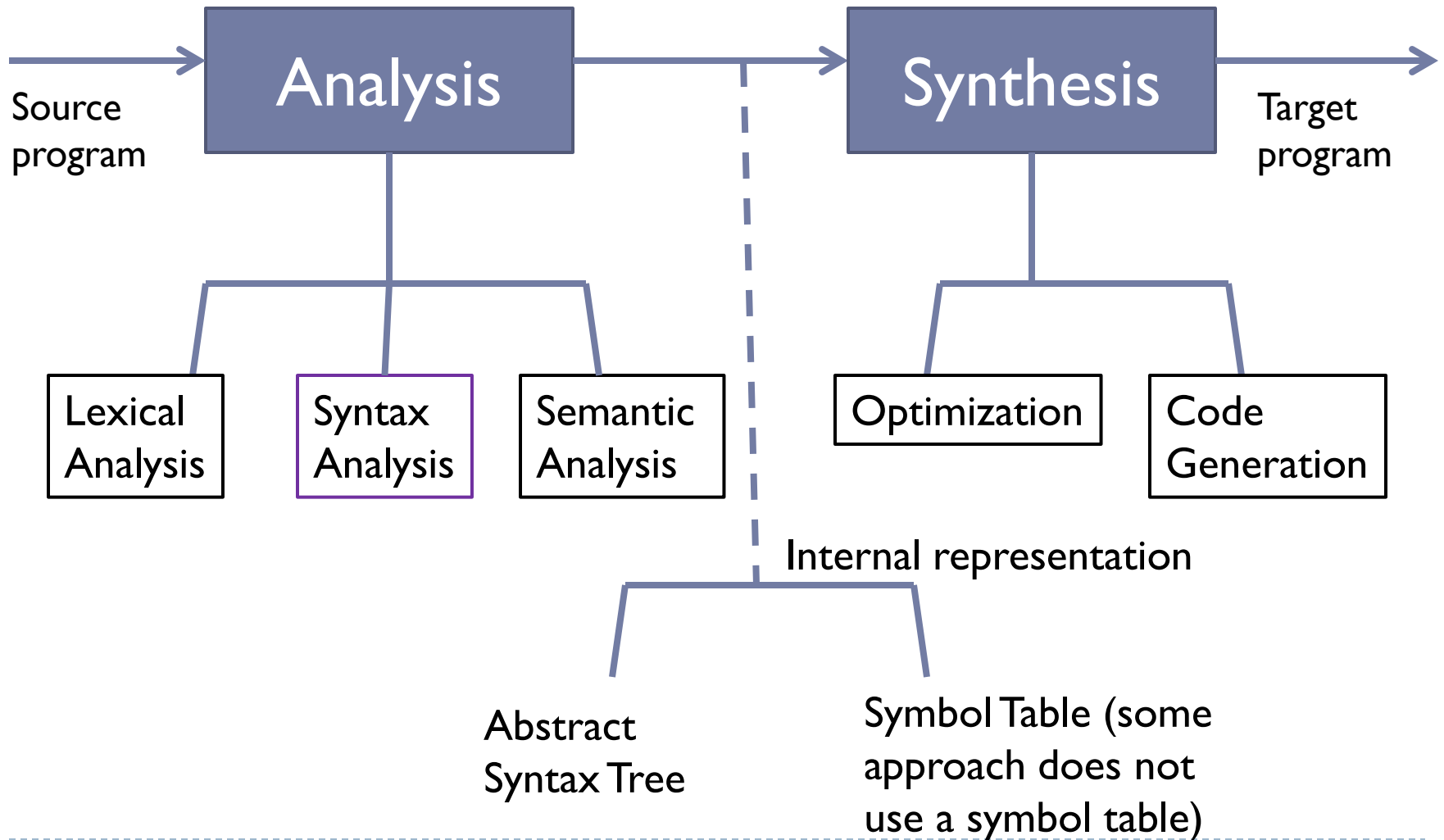
## ▶ Main references:

- ▶ Crafting a Compiler, by C. N. Fischer, R. K. Cytron, R. J. LeBlanc, Pearson, 2010, ISBN-13: 978-0-13-801785-9
- ▶ Compilers: Principles, Techniques & Tools, 2<sup>nd</sup> Ed, by A. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Pearson, 2006, ISBN-13: 978-1-292-02434-9.

## ▶ Other reference:

- ▶ Modern Compiler Implementation in Java, 2<sup>nd</sup> Ed, by A. Appel, J. Palsberg, Cambridge University Press, 2004, ISBN 0-521-82060-X.

# The Structure of a Compiler



# Lexical Analysis

---

- ▶ The first step: recognize words.

- Smallest unit above letters

This book is the main reference.

- ▶ Lexical analyzer divides program text into “words” or “tokens”

If x == y then z = 1; else z = 2;

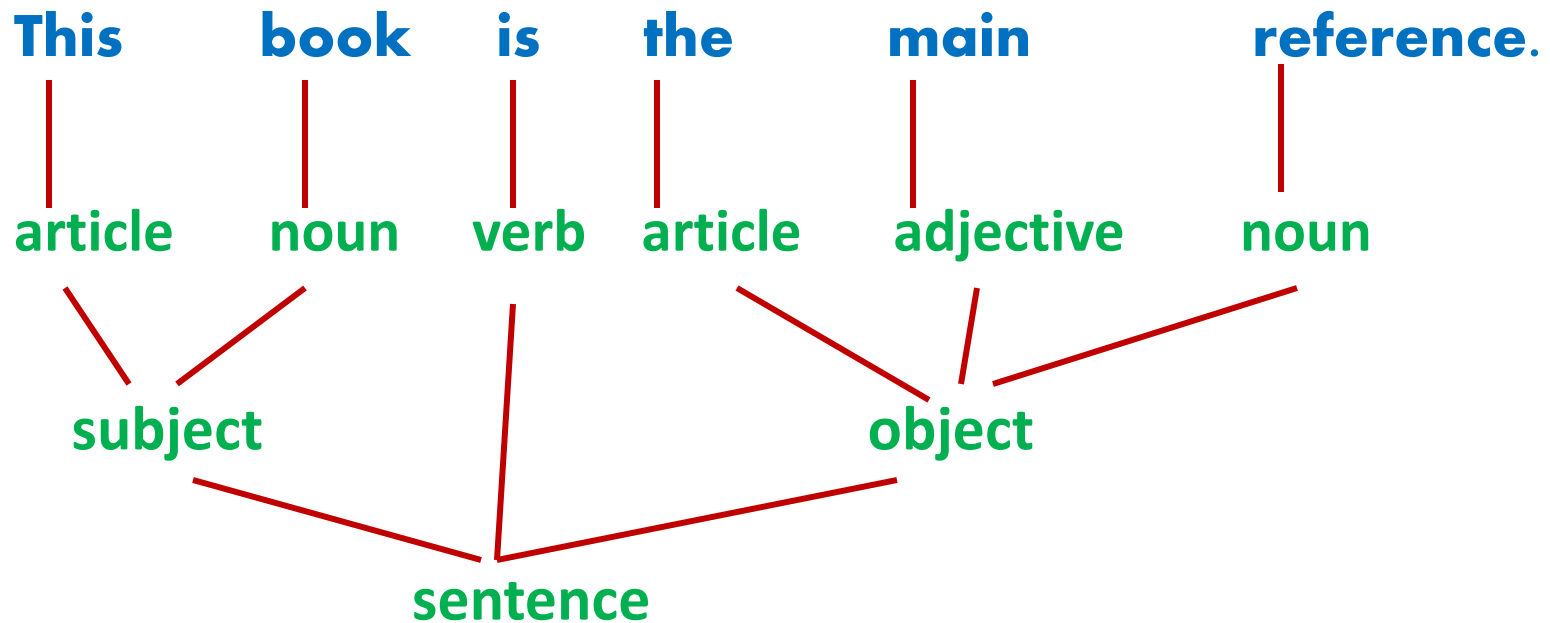
- ▶ Tokens:

IF id EQ id THEN id ASSIGN int literal ...



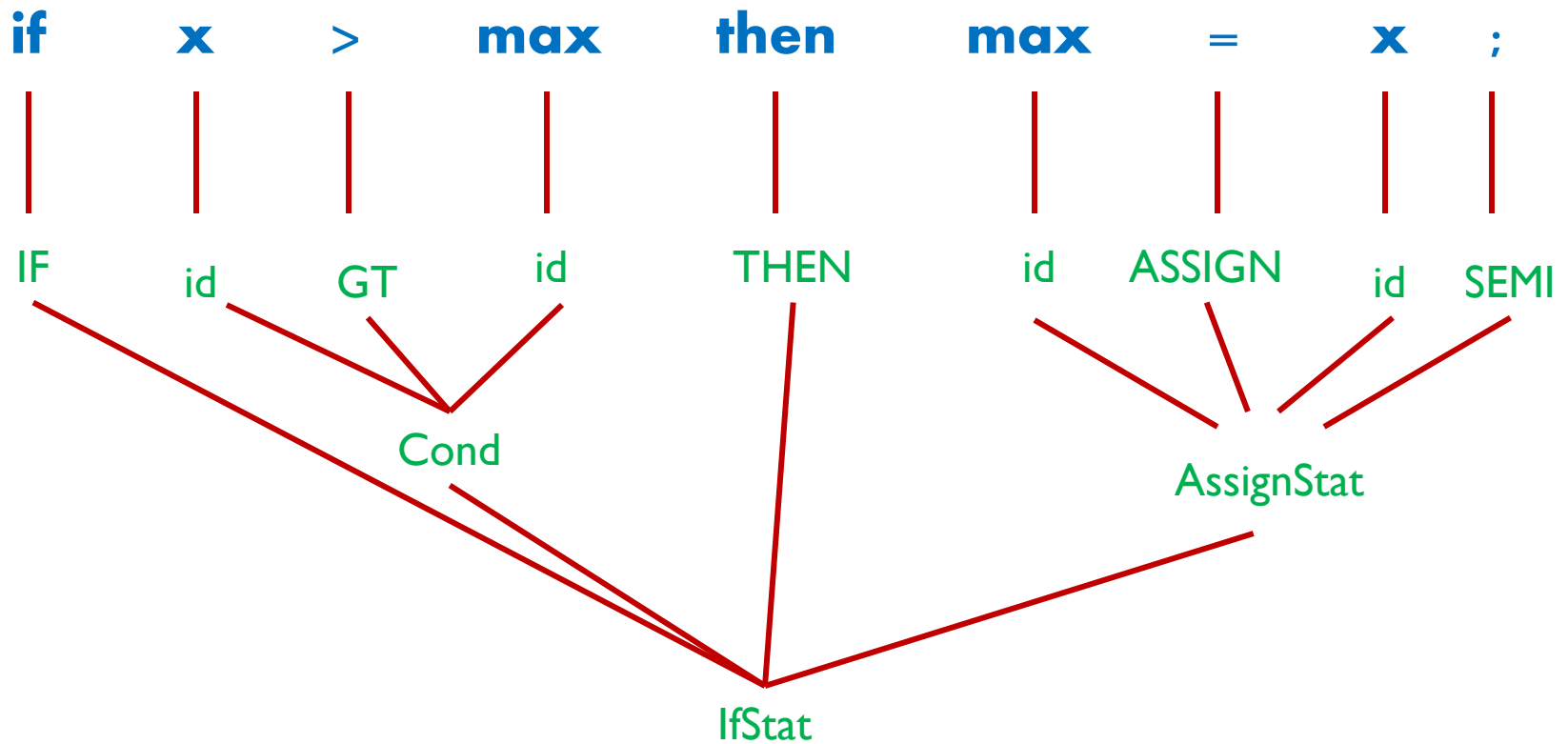
# Syntax Analysis

- ▶ Once words are understood, the next step is to understand sentence structure (and catch syntax errors)
- ▶ Syntax analysis (Parsing) = Diagramming Sentences
  - ▶ The diagram is a tree



# Syntax Analysis

- ▶ Parsing a program is the same



# Semantic Analysis

---

- ▶ Once sentence structure is understood, we can try to understand “meaning”
  - But meaning is too hard for compilers
- ▶ Compilers perform limited analysis to catch inconsistencies

For example, can you detect what are wrong below?

```
{  int a = 0;  
    float a = -0.5;  
    cout << a;  
    b = a;  
    a = “hello”;  
}
```

# Optimization

---

- ▶ No strong counterpart in English, but akin to editing
- ▶ Automatically modify programs so that they
  - Run faster
  - Use less memory
  - In general, conserve some resources

For example,


```
t = z * 2;  
if (t > y) {  
    z = z * 2;  
    ..... }  
}
```

# Code Generation

---

- ▶ Produces assembly code (usually)
- ▶ A translation into another language
  - Analogous to human translation

For example,

$Y = X * 10.0;$  

```
LDF  R2, X
MUL  R2, #10.0
STF  Y, R2
```

# Some Review Questions/tasks

---

1. 'google' the term 'machine code'.
2. 'google' the term 'assembly language'.
3. What does a compiler do?
4. What does an interpreter do?
5. What is the structure of a compiler and what are the main tasks involved?

# AUTODESK LABS: DESIGNSCRIPT

Automate your workflow with powerful scripting.



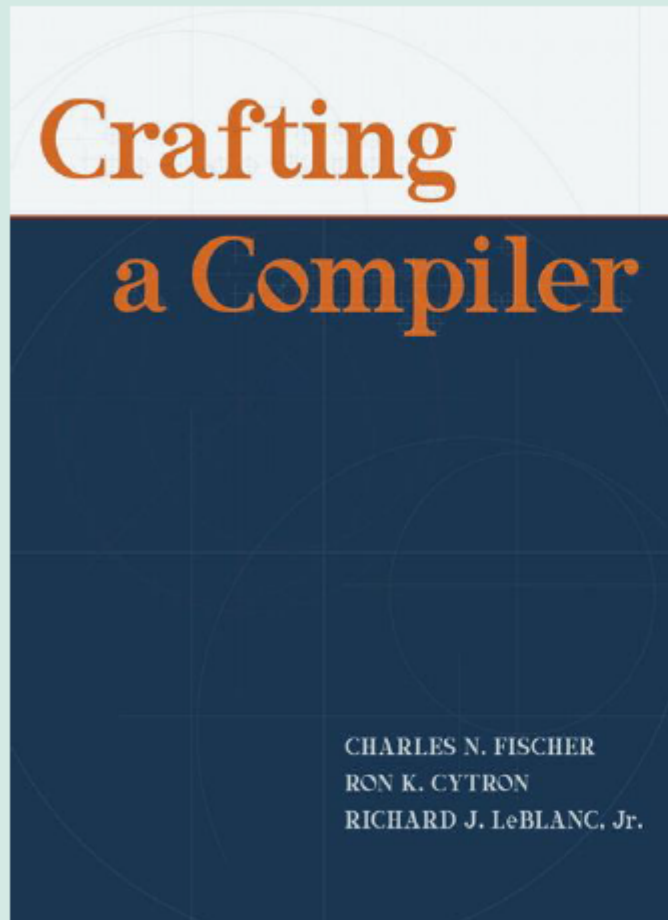
DesignScript is a unique language. It is intended to help designers build and analyze complex geometric models that would be difficult to model with interactive techniques. ...

DesignScript introduces the distinction between a generative description of a design (as a script) and the resulting generated model. The designer no longer directly models the resulting design: instead he develops a script whose execution generates the model. ...

A hand crafted parser for DesignScript that will perform better than the current auto-generated parser is desired. ...

[back](#)

# Get it at NTU bookstore!

[back](#)

## CZ3007 Compiler Techniques

Textbook author: Charles N. Fischer

Textbook title: Crafting A Compiler

Textbook edition: 1

ISBN: 9780138017859

***Grab a friend to  
enjoy \$20 off***

for every two copies of the  
same title purchased.