Ambo University Woliso Campus

Department of Computer Science

Semester II, 2020

Compiler Design
Credits: 3

Instructor: Mr. Yoobsan B

COURSE OBJECTIVES:

Upon completion of this course, students will have gained knowledge of compiler design and construction concepts and to:

- Introduce the major concept areas of language translation and compiler design.
- Understand the phases of the compilation process and be able to describe the purpose and implementation approach of each phase.
- Know how to use compiler construction tools, such as generators of scanners and parsers
- Be able to define LL(1), LR(1), and SLR(1) grammars with parsing techniques
- Design a compiler for a simple programming language; and Implement a compiler based on its design…

Text book

- Compilers: Principles, techniques and tools by *Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman*

Reference book

- Compiler construction : Principles and practice; *Kenneth C.Louden*

Course outline

1. Introduction

- Analysis and synthesis in a compilation
- Various phases in a compilation
- Grouping of phases
- Major data and structures in a compiler
- Compiler construction tools

2. Lexical analysis and Lex

- Token, pattern, lexeme
- Attributes of a token
- Errors
- Specification of tokens using regular expressions
- Regular expression for programming language tokens
- Recognizing tokens using transition diagrams
- Design of lexical analyzer
- Construction and simulation of NFA and DFA
- Conversion from RE – NFA – DFA
- Lex scanner generator

3. Syntax analysis and Yacc

- Role of a parser
- Context Free Grammar
- Derivation, parse tree, ambiguity, left recursion, left factoring
- Syntax analysis
- Syntax error handling
- Top down parsing
- Recursive decent parsing
- Non recursive predictive parsing
- Bottom up parsing
- LR(k) parsing
- Shift reduce parsing
- Construction of SLR parsing table
- Yacc parser generator

4. Syntax directed translation

- Syntax directed definitions
- Dependency graph and evaluation order
- S-attributed definitions
- Bottom-up evaluation
- Top-down evaluation
- L-attributed definitions

5. Type checking

- Type systems
- Specifications of a type checker
- A simple language example
- Equivalence of types
- Type conversion

6. Intermediate code generation

- Intermediate languages
- Types of three address statements
- Syntax directed translation into three address code
- Implementation of three address statements
- Translation scheme to generate three address code
- Addressing array elements

7. Code generation and optimization

- Issues in the design of a code generator
- A simple code generation algorithm
- Memory management
- Instruction selection
- Register allocation

Evaluation methods:

- ✓ Quizzes 10%
- ✓ Lab 15%
- ✓ Assignment/projects 15%
- ✓ Mid/Tests 20%
- ✓ Final 40%