Fundamentals of Compiler Design

Syllabus

LIST OF TOPICS

-Introduction to complier

- Types of languages and grammars

- Word analysis and correction of word errors

- Syntax analysis

- Top-down decomposition methods

- Low-level decomposition methods

- The primacy of the operator

- Simple precedence

- analysis LR(1) including SLR(1), LALR (1), and CLR (1)

- Semantic Analysis

- Symptom table management

- Run-time memory allocation methods

- Code Generation

- Payment and code optimization

- Automatic production of compilers

OUTCOMES

-Familiarity with compiler components and their implementation techniques

- Understanding the execution of programming language commands

- Skills in producing optimal programming and fixing programming errors

-Introduction and application of automated tools in production of compiler

REQUIRED SOFTWARES

An appropriate web programming frameworks-ANTLR

ASSESSMENT

Home works :15%

projects :15%

exams :70%

GOALS

Designing and building compilers is one of the basic concepts of computer science. Although the methods of creating compilers are slightly varied, they can be used to construct interpreters and translators of a wide variety of languages and machines. In this lesson, the topic of creating compilers is introduced by describing the main components of a compiler, their tasks, and their relevance.

After a basic introduction to compiler components and types of grammars, various translation steps such as lexical, syntactic and semantic analysis and code generation and processing are described.

REFERENCE BOOKS

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman , Compilers:Principles, Techniques,and Tools.

Second Edition, Boston: Addison-Wesley, 2007.