Course-Plan
Spring 2018

School	**Engineering**
Department	**Computer Science and Engineering**
Course Code	**CO 308**
Course Name	**Compiler Design**
Instructor	**Utpal Sharma**

1. **Abstract**: The course CO 308 Compiler Design deals with the topic of compiler design for students who are well aware of the structure and semantics of programming languages. A compiler is a software tool that translates programs written in high level language to a specific machine language. The challenges in this task are interesting and have been well studied over time. The task is considered in phases and suitable modelling and mechanisms are employed. For some of these sub-tasks useful tools are available. This course covers these aspects with adequate practical exercises.

2. **Objective:**

Module	Topic	Learning Objectives
1	Introduction	To get an overall idea of Languages and grammar and the phases of a compiler.
2	Lexical Analysis	To understand the different ways in which lexical units (tokens) can be identified in an input program.
3.	Syntax Analysis	To learn the formal representation of program language syntax, and the different parsing techniques available.
4.	Intermediate code-generation	To understand the need and ways of suitable representation of the output of the syntax analysis (parsing).
5.	Semantic Analysis	To understand the how different syntactic constructs have different semantics, and how these can be dealt with for the purpose of translation.
6.	Code Optimization	To understand the scope of optimization in producing the machine language instruction sequence for a given input HLL program.
7.	Code-generation	To understand the run-time environments, translation of language constructs, scope for optimization specific to a given machine, and algorithms for code-generation.
8.	Error Handling	To understand the different types of errors in an input program and how the exercise of translation can be carried out in presence of these.

3. **Prerequisites of the course:**
   Knowledge of programming, computer organisation and architecture, and formal languages and automata.

4. **Course outline (See Syllabus)**

Text Book:

Aho, A.V., Sethi, and Ullman J.D: C*ompiler Design*. Pearson Education, 2009

Reference Books:

- **Dhandhere,** *System programming and operating systems***, Tata McGraw Hill.**
- **Jean-Paul Tremblay and Paul G. Sorrenson.** *The Theory and Practice of Compiler Writing*, McGraw Hill Book Co.

5. **(a)Time-Plan**

Tentative Lectures	Topics
1-2	Characteristics of HLL, overview of a compiler's task, phases of compilation task
3	Lexical Analysis (refer to coverage in System Software/System Programming)
4-5	An operator precedence parsing scheme without considering a grammar, advantages and limitations. *Exercise*: Implement a desk calculator using operator precedence parsing.
6-8	CFG description of programming languages; parse trees, derivation sequences, ambiguity, top-down and bottom-up parsing *Exercise*: Familiarise with grammar representation in C.
9,10	Automatic creation of an operator precedence table from a CFG *Exercise*: Implement algorithms for computing LEADING and TRAILING sets, and construction of operator precedence table
11-14	Top down Parsing: Recursive descent parser; predictive parser; construction of a LL(1) parsing table; LL(K) grammar *Exercise*: Implement the FIRST and FOLLOW algorithms required for LL(1) parsing table construction
15-22	Bottom-up parsing: LR parsing; SLR parsing table creation; CLR and LALR parsing table creation *Exercise*: 1. Implement the SLR parsing table creation method. 2. Learn use of tool *bison.*
23-25	Intermediate code generation: Syntax directed translation; Intermediate code formats; *Exercise*: Generate intermediate code using semantic actions in bison program.
26,27	Storage allocation and Symbol table *Exercise*: Create symbol table using semantic actions in bison program.
28-33	Code optimization: Basic block, flow-graphs, loop detection, loop optimization, data flow analysis
34-36	Code generation: Efficient use of registers, instructions. *Exercise*: Produce m/c code from intermediate code.
37-39	Error Handling: Different types of error, techniques for error handling

Note:

Laboratory exercises:

Laboratory exercises are indicated against relevant topics in the above plan. Evaluation of laboratory works will be covered within Test IV mentioned in the evaluation plan.

(b) Evaluation plan

Test I	25
Test II (Major I)	40
Test II (Lab Assignments)	25
End Term	60
Total	**150**

6. Pedagogy :

Teaching-learning methods to be used

Lecture and Discussion
Term assignment
Class assignments

7. Expected outcome: Towards the end of the course the student would understand the task of translation of high level language programs into machine language programs, the formal modelling of the problem at various stages and techniques and tools for carrying out the task. This course is also expected to provide the more keen students the skills to model some other problems in a suitable ways so as to be able to use some tools and techniques that are used in the compilers.