SECOND SEMESTER 2015-2016

Course Handout (Part II)

Date:14-01-2016

In addition to part I (General Handout for all courses appended to the time table) this portion gives further specific details regarding the course.

Course No: **IS F342**
Course Title: **Compiler Design**
Instructor-in-charge: Vandana Agarwal (vandana@pilani.bits-pilani.ac.in)
Instructor: Prerna Kaushik (p2013192@pilani.bits-pilani.ac.in)
Course Website: **http://nalanda.bits-pilani.ac.in/**

1. Objective

To expose the students to a class of special algorithms to process languages to translate high level user programs to low level assembly code. This includes algorithms and data structures handling both front end and back end of the compiler.

2. Scope

This course is an introductory course to compiler construction. In this course, students will learn the important basic elements of compilation to use the techniques effectively to design and build a working compiler. Topics include lexical analysis, parsing techniques, syntax directed translation, symbol table, intermediate code generation, data flow analysis, code generation, code optimization, error detection and recovery. Students will also participate in small teams in developing the building blocks of a compiler through compiler project.

3. Books

3.a. Text Book

T1. **Aho, Lam, Sethi and Ullman**.
Compilers Principles, Techniques, and Tools.
Pearson Education. Low Price Edition. 2004

3.b. Reference Books

R1. **Ravi Sethi**, **Programming Languages - Concepts and Constructs**
Pearson Education. Low Price Edition. 2003
R2. Andrew Appel, Modern Compiler Implementation in Java.
Cambridge University Press. (Foundation Books, New Delhi.) Rev. Ed. 2000.

4. Course Plan

4.a. Modules and Learning Objectives

Module	Title	Learning Objective(s)
C1	**Introduction to Compilers**	To understand the context and use of a compiler.
C2	**Front End of a Compiler**	To understand the implementation of a front end of a compiler - scanning, parsing and semantic analysis.
C3	**Back End of a Compiler**	To understand the implementation of a back end of a compiler - run time environments, Code Generation and Register allocation.
C4	**Special aspects of compilers and runtime**	To understand some special aspects of compilers and runtime such as code optimization, garbage collection etc.

4.b. Lecture Schedule

Lec.	Topic	Module
1	Overview of the course	C1
2-3	Lexical Analysis-Tokens, patterns, lexemes, lexical errors, input buffering, regular expressions, recognition of tokens, Transition diagrams, recognition of reserved words and identifiers, transition diagram based lexical analyzer, look ahead operator, Lexical-analyzer generator	C2
4-5	Syntax Analysis: Parser, context free grammars, syntax errors and recovery, parse trees, derivations, ambiguity, syntactic correctness, Recursive Descent Parsing, left recursion, left factoring and ambiguity.	C2
6-8	Syntax Analysis: Top Down Parsing- first and follow sets, LL(1) Grammars, construction of predictive parsing table, Non Recursive Predictive parsing, error recovery, constructing Parse Tree	C2
9-11	Syntax Analysis: Bottom Up Parsing- LR parsing, reductions, handle pruning, shift-reduce parsing, conflicts, Simple LR parsing, Items and LR(0) Automaton, LR parsing algorithm, SLR parsing tables, more powerful LR parsers, parser generators.	C2
12-14	Syntax directed translation, inherited and synthesized attributes, evaluation order, dependency graphs, evaluation of attributes construction of Abstract Syntax Tree (AST)	C2

15	Symbol table, handling scope in the symbol table, data structure for symbol table, linking AST with symbol table, symbol table implementation of the function parameters.	C2
16	Type Checking and Type Inferencing- type expressions, type equivalence	C2
17-18	Intermediate Representation; Intermediate Language , three address code, quadruples, triples, semantic rules to generate intermediate code	C3
19-20	Code Generation-target language, program and instruction cost, addresses in the target code, run time addresses for names, basic blocks, next use information, flow graphs, optimization of basic blocks, register and address descriptors, code generation algorithm.	C3
21-22	Data Flow Equations. Liveness analysis.	C3
23	Register Allocation	C3
24-25	Code optimization- Peephole optimization, redundant code elimination, flow of control optimizations.	C3
26	Garbage Collection	C4

5. Evaluation Scheme

5.a. Major Components

Component	Mode	Duration	Date	Weight
Mid Semester Test	Closed Book	90min	15/3 9:00 - 10:30 AM	**25%**
Compiler Project	Lab (Take home)	8 weeks		**35%**
Comprehensive	Partially Closed Book/Open book	3 hours	5/5 FN	**40%**

5.b.Compiler Project:

- The project will be implemented by students (in a team of 2) in **TWO** stages/ components.
- Timely submitted components will be evaluated individually through a viva-voce, demonstration or may be evaluated offline.
- Only originally created code will be evaluated. Both students of the team submitting plagiarized code will be penalized as per the malpractice regulations.[refer 5(c)]
- Compiler components are to be completed in time with no postponements.

- Each team gets only ONE lifeline over the semester. A lifeline allows the team to submit the code at most 24 hours late without any penalty.
- Once the lifeline has been used up, 24 hour delays may be permitted for submission of the second stage at the penalty of 25% weight for that component. No submissions are allowed after 24 hours from deadline.
- Each stage should be developed incrementally.
- If a team of students does not complete/ get fully functional code for larger set of features or operations as expected at any stage, the team will be allowed to submit the code that works correctly for less features, operations etc. The marks will be allotted partially as per the criteria. However, the partially working code of an previous stage may be a limitation for the later stages, which students may have to work out to be able to have the later stage code get appropriate input from the previous stage.

Compiler Project Components

Stage	Title	Duration	Weight
1	Lexical Analyzer, Predictive Parser, syntax Analysis.	4 weeks	15%
2	Abstract Syntax Tree Generation, Symbol table, Semantic Analysis, Code Generation and Compiler Integration	5 weeks	20%

5.c. Malpractice Regulations:

The following regulations are supplementary to BITS-wide policies regarding malpractices:

- Any student or team of students found involved in malpractices in working out assignments / projects will be awarded a zero for that assignment / project and will be blacklisted. Note that the entire project component will be awarded zero, irrespective of the stage at which the mal practice is found.
- Any student or team of students found repeatedly – more than once across all courses – involved in malpractices will be reported to the Disciplinary Committee for further action. This will be in addition to the sanction mentioned above.
- A malpractice, in this context, will include but not be limited to:
  - submitting some other student's / team's solution(s) as one's own;
  - copying some other student's / team's data or code or other forms of a solution;
  - seeing some other student's / team's data or code or other forms of a solution;
  - permitting some other student / team to see or to copy or to submit one's own solution;
  - OR other equivalent forms of plagiarism wherein the student or team does not work out the solution and/or uses some other solution or part thereof (such as downloading it from the web).

4. The degree of malpractice (the size of the solution involved or the number of students involved) will not be considered as mitigating evidence. Failure on the part of instructor(s) to detect mal-practice at or before the time of evaluation may not prevent sanctions later on.

6. Notices: All notices concerning this course will be put on the **CSIS notice board** OR the course website as appropriate.

7. Makeup Policy:

- **Permission of the Instructor-in-Charge is required** to take a make-up
- **Make-up applications must be given to the Instructor-in-charge personally.**
- ***A make-up test shall be granted only in genuine cases where - in the Instructor's judgment - the student would be physically unable to appear for the test.***
- In case of an unanticipated illness preventing a student from appearing for a test, the student must present a Medical Certificate from BITS hospital.
- In case of unanticipated absence for a test due to a trip out of Pilani, the student must present a letter from his/her Warden or the Chief Warden certifying such absence and the reason(s).
- Requests for make-up for the comprehensive examination – under any circumstances – can only be made to Dean, Instruction Division.

Instructor-in-charge
IS F342