SECOND SEMESTER 2022-2023

Course Handout (Part II)

Date: 19-01-2023

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 Number:	CS F363
Course Title:	Compiler Construction
Instructor-in-charge:	Vandana Agarwal (vandana@pilani.bits-pilani.ac.in)
Instructor:	Shashank Gupta (shashank.gupta@pilani.bits-pilani.ac.in)
Course Website:	http://nalanda.bits-pilani.ac.in/

1. Objective

The students will be introduced to special algorithms for processing 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 of the course

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 will 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 develop the building blocks of a compiler through compiler project.

3. Pre-requisite courses

The course on Compiler Construction uses the concepts learnt in earlier courses such as *'Theory of Computation', 'Data Structures and Algorithms' and 'Principles of Programming Languages'* which are also the pre-requisite courses done by the students. Students will be expected to have sufficient understanding of the concepts of finite automata theory e.g. DFA (Deterministic Finite Automata), NFA (Non-deterministic Finite Automata), State Minimization Algorithm, CFG (Context Free Grammar), parse tree, ambiguity, push down automata etc. Also, each student should have a prior knowledge of data structures and experience in implementing linked list, stacks, trees, hash table, dynamic arrays etc. The knowledge of various language constructs and features such as statements, variables, activation records, scoping, variable binding etc. will be required to understand the complex implementations needed in compiler development. The above three courses are used rigorously in compiler design and construction and we will be using them very often, which should not be misunderstood as the repetition of any of the three pre-requisite courses.

4. Books

Text Book

T1 Compilers Principles, Techniques, and Tools.
Authors: Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffery D. Ullman
Publisher: Pearson Education. Second Edition.

Reference Books

R1 Programming Languages - Concepts and Constructs
Author: Ravi Sethi
Publisher: Pearson Education.

R2 Introduction to Algorithms
Authors: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
Publisher: Prentice Hall of India

R3 Modern Compiler Implementation in C.
Author: Andrew W. Appel
Publisher: Cambridge University Press. (Foundation Books, New Delhi.) .

R4 Modern Compiler Implementation in Java.
Author: Andrew W. Appel
Publisher: Cambridge University Press. (Foundation Books, New Delhi.) .

R5 Concepts in Programming Languages
Author: Robert W. Sebesta
Publisher: Pearson Education Inc.

5. Course Plan

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.

Lecture Schedule

Lec.	Topic	**Module**
1-2	Overview of the course-language processors-compiler and interpreters, Phases of compiler, need for different compilers, attributes of a good compiler, source and target code, a language processing system, structure of a compiler. Course administration overview-evaluation components, project details etc.	C1

3-4	Lexical Analysis-Tokens, patterns, lexemes, lexical errors, input buffering, regular expressions, recognition of tokens, Transition diagrams, recognition of reserved words and identifiers, architecture of a transition diagram based lexical analyzer, look ahead operator, Pattern matching based on NFA, DFA for lexical analyzer, Optimization of DFA based pattern matcher, State minimization in lexical analyzers, DFA simulation.	C2
5-6	Syntax Analysis: Role of a parser, representative grammars, syntax error handling, error recovery strategies, CFG, parse trees, derivations, ambiguity, syntactic correctness, parsing process, Recursive Descent Parsing, left recursion elimination, left factoring.	C2
7-9	Syntax Analysis: Top Down Parsing- first and follow sets, LL(1) Grammars, construction of predictive parsing table, Non Recursive Predictive parsing, error recovery in predictive parsing.	C2
10-13	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.	C2
14-16	Syntax directed translation, inherited and synthesized attributes, evaluation order, dependency graphs, evaluation of attributes, construction of Abstract Syntax Tree (AST)	C2
17-18	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, variety of semantic checks using symbol table. Type Checking and Type Inferencing- type expressions, type equivalence	C2
19-21	Intermediate Representation; Intermediate Language , three address code, quadruples, triples, semantic rules to generate intermediate code for various constructs.	C3
22-24	Code Generation-target language, program and instruction cost, addresses in the target code, run time addresses for names, register and address descriptors, code generation algorithm.	C3
25-26	Run Time Environments: Storage organization, Stack allocation of space, contents of an activation record, memory models, calling sequences, variable length data on the stack, access to non-local data on the stack, data access without nested scopes, issues with nested procedures, nesting depth, access links.	C4

27-29	Code Optimization-Basic blocks, next use information, flow graphs, optimization of basic blocks, Liveness analysis.	C3
30-32	Register Allocation, Instruction selection	C3
33-34	Code optimization- Peephole optimization, redundant code elimination, flow of control optimizations.	C3
35-36	Machine Independent Optimizations-Global Common Sub-expressions, copy propagation, dead code elimination, induction variables and reduction in strength etc.	C4
37-38	Garbage Collection-Mark and sweep algorithm, reference counting algorithm	C4
39-40	Discussion on few existing compilers	-

6. Evaluation Scheme: The major components are as follows. Total marks will be 300.

Major Components

S. No.	Component		Mode	Duration	Date	Weight
1	Mid Semester Test		Closed Book	90 min	TBA (visit AUGSD website)	**20%**
2	Compiler Project (50% weight)	Code development (Team-based)	Take home, two stages of submission, to be done in teams of 3-5 students.	8 weeks	TBA (visit Nalanda course page regularly)	**35%**
		Online exam (Individual)	Open book	3 hours	TBA (visit Nalanda course page regularly)	**15%**
3	Comprehensive Exam		Closed Book	3 hours	TBA (visit AUGSD website)	**30%**

7. Compiler Project

(a) Code Development

- The project will be implemented in **teams** of 3-5 students in **two** stages/components. The maximum team size cannot exceed five students in it. However, it is observed that large teams tend to find difficulty in maintaining trust and compatibility among the members. It is advised that the students must form teams with utmost care such that they have good team dynamics among the team members.

The purpose of allowing students to work in teams is to encourage learning by working together, sharing the knowledge and creative thinking among students. Students in teams are encouraged to discuss any concepts and implementation issues among themselves to constantly grow intellectually.

- The students will be asked to form the teams on their own and register their team members' names with us. The teams will not be allowed for any change of names of its members during the complete semester. Students are advised to form teams on their own, based on their mutual understanding, trust and compatibility. The teams once formed will be expected to work towards the code development with similar rigor irrespective of the team size.
- Timely submitted components will be evaluated through viva-voce, demonstration or may be evaluated offline.
- Only originally created code will be evaluated. Students submitting plagiarized code will be penalized as per the malpractice regulations.
- Compiler components are to be completed in time with no postponements.
- Each team gets only **one** lifeline over the semester. A lifeline allows teams to submit the code at most 24 hours late without any penalty.
- Once the lifeline has been used up, 24 hours delay may be permitted for submission of the second stage at the penalty of 25% weight for that component. No submission is allowed after 24 hours from deadline.
- Each stage should be developed incrementally.
- If students in a team do not complete/get fully functional code for larger set of features or operations as expected at any stage, the students 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 a previous stage may be a limitation for the later stage and for online exam as well. The stage 1 code will have to be completed by the students to be able to work on the second stage.
- All members of a team indulged in submitting plagiarized code at any stage will be penalized equally irrespective of who did what. It is advisable that all members of a team work together, motivating each other for genuine work and create good team dynamics among the team members. Interacting among team members and working together will prevent the students from getting plagiarized piece of code from any team member.
- The submitted code will be subjected for plagiarism check and the defaulters will be informed accordingly.
- The evaluation will be done only for teams whose code will be free from any degree and form of plagiarism. Team members will be awarded same marks for their compiler code developed by their team for project component. However, individual efforts put in by each student in the complete learning process and code development will be rewarded through individual performance in the separate online exam as discussed below.
- The stage 1 and stage 2 marks will be uploaded within a week after submission.

(b) <u>Online exam</u>

- The online exam will be based on compiler code submitted by the teams as above. In this exam, each individual student will further implement the asked features, constructs and semantics etc. to produce code in assembly language. The purpose of the online exam is to evaluate an **individual** student's performance for his/her efforts put in building the compiler during the code development stages and the knowledge gained thereafter.
- The online exam will be conducted in IPC labs on any day (preferably Sunday or any holiday) in the third or fourth week of April 2023 (Tentative date April 23, 2023). The duration of the exam will be of 3 hours. Strictly no makeup request will be entertained for online examination.

- All students of teams who submitted plagiarized code will not be able to appear for the online exam irrespective of their individual involvement in plagiarism. Therefore, it is the responsibility of each individual student within the team to ensure that no team partner contributes plagiarized piece of code for their code submission.
- Each individual student, whose team submitted the code free from any degree and form of plagiarism, will be able to download before the start of the exam his/ her compiler code submitted by his/ her own team.

8. Code development stages

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

9. Professional Behavior

Students are expected to display professional behavior in the course. The students are expected to attend lectures, participate in class discussions, read text and reference books regularly to gain in-depth understanding of the concepts, interact with the teacher frequently during the class or during chamber consultation hour, solve problems given as homework, show interest in the subject and feel enthusiastic about the overall learning. Students must inform the instructor about his/her absence from the class beforehand by sending an email. Students not attending classes regularly and missing more than 5-6 lectures during the course will be seen as professional misconduct. In an environment of attendance not being compulsory, maintaining freedom and flexibility, it is expected that the students feel responsible professionally and attend classes regularly.

10. Malpractice Regulations

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

- 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 malpractice is found.
- 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 solution(s) as one's own;
  - copying some other student's data or code or other forms of a solution;
  - seeing some other student's data or code or other forms of a solution;
  - permitting some other student to see or to copy or to submit one's own solution;
  - OR other equivalent forms of plagiarism wherein the student 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 malpractice at or before the time of evaluation may not prevent sanctions later on.

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

12. Chamber Consultation: *Chamber number : 6121-Z (Vandana), 6120-G (Shashank)*
Schedule: To be announced on Nalanda.

13. 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, AUGS-AGSR Division.

Instructor-in-charge

CS F363