# FORMAN CHRISTIAN COLLEGE UNIVERSITY
## COMP451: Compiler Construction (2+2 Credit Hrs)
## Course Outline and Lesson Plan

**Instructor Information:**
**Name:** M. Rauf Butt
**Contact:** raufbutt@fccollege.edu.pk
**Office:** S-214
**Office Hours:** TBD

**Pre Requisites:**

- Data Structures & Algorithms
- Theory of Automata

**Course Material:**

1. Lab/Class Activity Handouts
2. Class Handout
3. Video Lectures

**Text Books:**

1. Compilers: Principles, Techniques, and Tools, second edition; Aho, Lam, Seithi, and Ullman
2. Programming in C; By Dennis Richie
3. Flex & Bison; By John Levine

**Course Objectives:**

Differentiate between different levels of programming languages. Understand the role of front-end and back-end of a compiler. Recognize different types of grammars. Understand and define grammars in BNF, syntax diagrams, regular expressions. Define tokens using the notation of regular expressions. Convert regular expressions into finite automata. Implement a lexical analyzer. Define a programming language syntax using a CFG. Construct a parse tree for a given program. Differentiate between top-down and bottom-up parsing strategies. Understand LL (k) and LR (k) grammars. Write a top-down parser using recursive-descent and LL (1) parsing methods. Understand simple-precedence, operator precedence and SLR parsing methods. Understand semantic analysis (type checking, scope checking etc.) Understand various types of runtime environments. Understand code generation techniques. Understand code optimization techniques.

## Course Learning Outcomes (CLOs)

| CLO's | Description | Level |
|---|---|---|
| CLO:1 | Describe the architecture, and functions of different components of a compiler | C1 (Remember) |
| CLO:2 | Describe how a program gets executed and what are different programs other than compiler that help execute the program. | C2 (Understand) |
| CLO:3 | Application of formal notations to define a programming language | C3 (Apply) |
| CLO:4 | Design and implement different segments of lexical and syntax analyzer using an appropriate programming language. | C3 (Apply) |
| CLO:5 | Integrate different smaller segments and formulate a complete working lexical and syntax analyzer. | C3 (Apply) |
| CLO:6 | Differentiate and compare open source compiler, interpreter and cross compilers available. | C4 (Analyze) |

## Mapping of CLO's to PLO's

| PLOs | CLO:1 | CLO:2 | CLO:3 | CLO:4 | CLO:5 | CLO:6 |
|---|---|---|---|---|---|---|
| Computing Knowledge | √ | √ | √ | | | |
| Problem Analysis | | | | √ | | √ |
| Design and development of solutions | | | | √ | √ | |
| Investigation | | | | | | √ |
| Modern Tool Usage | | | √ | √ | √ | |

| Week | Theory Session (1 Hr 50 Min) | Lab Session (1 Hr 50 Min) |
|---|---|---|
| 1 | • Introduction to the course.<br>• Introduction to compilers<br>• Phases of compilation.<br>• An overview of phases of compilers and how these work.<br>• An example showing how a very simple line of code is passed through all the phases of compiler. | A primer in C Programming Language<br>• Basic program structure<br>• Variables and Pointers<br>• Programming constructs (loops, control structures)<br>• User defined functions |
| 2 | • Cousins of compilers<br>  ○ System programs that help compiler execute a program completely.<br>  ○ Looking into how a C program is expanded while execution.<br>  ○ Difference between .c, .i, .s, and .o files.<br>  ○ Explaining linking and loading.<br>  ○ Difference between static and dynamic linking. | A primer in C Programming Language …<br>• Arrays<br>• Strings<br>• Dynamic memory allocation.<br>• Structures in C |
| 3 | • Lexical Analysis<br>  ○ Working of a lexical analyzer<br>  ○ Formal definition of tokens, lexemes, patterns.<br>  ○ Identification of tokens<br>  ○ Regular Languages<br>  ○ Regular Expressions | **Quiz 1**<br>**Lab 1** |
| 4 | • Lexical Analysis …<br>  ○ Formal Languages<br>  ○ Lexical Specification<br>  ○ Finite Automata<br>  ○ Regular Expressions to NFA | **Lab 2** |
| 5 | • Lexical Analysis …<br>  ○ NFA to DFA<br>  ○ Implementation of DFA<br>  ○ Problem solving session for Lexical Analysis<br>• Syntax Analysis<br>  ○ Introduction to Parsing<br>  ○ Context Free Grammar<br>  ○ Left and Right Derivations | **Quiz 2**<br>Introduction to FLEX Programming<br>**Programming Assignment 1 Uploaded** |
| 6 | • Syntax Analysis …<br>  ○ Left and Right Derivations<br>  ▪ Problem solving session for left and right derivation<br>  ○ Ambiguity<br>  ▪ Why ambiguity? | **Lab 3** |

| | | |
|---|---|---|
| | ▪ How to eliminate it?<br>○ Abstract Syntax Trees<br>▪ Basic introduction<br>▪ How to create and traverse an AST | |
| 7 | • Syntax Analysis …<br>○ Recursive Descent Parsing<br>○ Left Recursion<br>○ Predictive Parsing<br>○ | **Lab 4**<br>**Quiz 3**<br>**Programming Assignment 1 Due** |
| 8 | • Syntax Analysis …<br>○ FIRST and FOLLOW | **Lab 5** |
| 9 | Revision Session | **Mid Exam (Course covered till end of week 8)** |
| 10 | • Syntax Analysis …<br>○ LL1 parsing table<br>○ Introduction to Bottom Up Parsers<br>○ Shift Reduce parsing | **Lab Exam (Tentative)** |
| 11 | • Syntax Analysis …<br>○ Operator Precedence parser<br>○ Introduction to Bison | **Lab 6**<br>**Programming Assignment 2 Uploaded** |
| 12 | • Syntax Analysis …<br>○ LR(0) Parser<br>○ SLR(1) Parser<br>○ LALR(1) Parser | **Lab 7**<br>**Quiz 4** |
| 13 | • Syntax Analysis …<br>○ CLR(1) Parser<br>• Ways to represent semantic rules:<br>○ Syntax Directed Definition<br>○ Syntax Directed Translation | **Lab 8**<br>**Programming Assignment 2 Due**<br>**Class Project Uploaded** |
| 14 | • Difference between SDD and SDT<br>• Types of SDD<br>○ S Attributed<br>○ L Attributed<br>• Examples of SDD<br>• Examples of SDT | **Lab 9**<br>**Quiz 5** |
| 15 | • Intermediate Code Generation<br>○ Why intermediate code<br>○ Types of Intermediate Code<br>▪ Syntax tree<br>▪ Three Address Code<br>○ Implementation of three address code<br>▪ Quadruple<br>▪ Triple<br>▪ Indirect Triple | **Lab 10** |

| | | |
|---|---|---|
| | ○ Three address code for flow of control statements<br>○ Three address code for arrays<br>• Three address code for case statements | |
| 16 | • Code Optimization<br>○ Detecting loops<br>○ Basic Blocks<br>○ Program flow graphs<br>○ DAG | **Class Project Due** |

Note that this outline is not carved on stone. Course staff / instructor reserves all rights to make appropriate changes as per needed.

**Assessment Criteria**

- In Class Quizzes 15%
- Labs 20%
- Mid Semester Exam 20%
- End Semester Exam (Comprehensive) 30%
- Programming Assignments / Home Work 7%
- Class Project 8%

**NOTE:**

- This is a lab course and we will conduct lab sessions almost every week.
- Labs will be conducted in class and hence only those students will perform lab who are present in the class.
- Students will have to prepare a report for every lab. Format of the lab will be uploaded on Moodle course page.
- We may have 4 to 6 quizzes. If number of quizzes is greater than 5, we may drop one quiz.
- Assignments/Home works will be uploaded and **MUST** be submitted within the deadline specified on handout.
- There will be **no retake for any instrument.**
- In case if any student under special circumstances is allowed to take entire course online, he/she will have to attempt the labs online within the given time frame.
- Online students (if any) will have to go for an online mid exam followed by a viva.
- More details will be provided in the introductory lecture during first week of this semester.
- Online students should feel free to ask any query via email or we can have an online zoom meeting.
- **Students are advised to attend all assigned lectures**. It is entirely the students' responsibility to recover any information or announcements presented in lectures from which they were absent.
- **It is mandatory to maintain a minimum of 75% attendance**.
- **In case if the attendance drops below the given threshold student will have to present written permission from the HoD to appear in the MID and or Final Exam**.
- **All work** that you submit in this course **must be your own**.
- **Unauthorized group efforts** are considered academic dishonesty.
- You may discuss homework in a general way with others, but you may not consult anyone else's written work. You are guilty of academic dishonesty if:
    - You **examine another's solution** to an assignment
    - You **allow another student to examine your solution** to an assignment
    - You fail **to take reasonable care to prevent another student from examining your solution** and that student does examine your solution.
- **Cheating, plagiarism and other forms of academic fraud** are taken very seriously. University Policy of plagiarism will be applicable in the case.
- **Attendance** does not carry any graded marks. However, be very cautious as we may have pop up quizzes in class.