**Mukesh Patel School of Technology Management and Engineering**

**Computer Engineering Department**

**Course Policy**

| **Program/Branch/Semester** | **:** | B.Tech Computer CSBS/ Sem V |
| --- | --- | --- |
| **Academic Year** | **:** | 2021-22 |
| **Course Code & Name** | **:** | Compiler Design |
| **Credit Details** | **:** | | L | T | P | C | | --- | --- | --- | --- | | 3 | 0 | 2 | 4 | |
| **Course Coordinator Faculty** | **:** | Prof. Ameyaa Biwalkar |
| **Contact No. & Email** | **:** | ​022-42334754  **ameyaa.biwalkar@nmims.edu** |
| **Office** | **:** | MPSTME building 3rd Floor (3 A Faculty area) |
| **Student Contact hours** | **:** | Tuesday (11am to 1 pm)  Friday (2 pm to 4 pm) |
| **Other Course Faculty members teaching this course** | **:** | **Course Faculty 1:** Prof. Asmita Marathe  **Contact No. & Email: asmitamarathe90@gmail.com**  **Office:** 91-8452828727 (Navi Mumbai Campus)  **Student Contact Hours:** Thursday (11 am to 2 pm)  Friday (12 pm to 2 pm) |
| **Course Faculty 2:** Dr. Navneet Pratap Singh  **Contact No. & Email:** **navneet.diat@gmail.com**  **Office:** 91-7389747504 (Indore campus)  **Student Contact hours:** Thursday (4 pm to 6 pm) | | |
| ***Queries by Emails are encouraged.*** | | |
| **Course link** | **:** | https://teams.microsoft.com/l/channel/19%3ac7185174cbca4e6a8e58b11f511b6406%40thread.tacv2/Compiler%2520Design?groupId=24cd8e7c-20c7-4240-9abd-7ffe199d683e&tenantId=d1f14348-f1b5-4a09-ac99-7ebf213cbc81 |

# Introduction to the Course

## Importance of the course

The subject Compiler Design is used to recognize the underlying formal models such as finite state automata, push-down automata and their connection to language definition through regular expressions and grammars and discuss the effectiveness of optimization.

## Objective of the Course

## The objective of the course is to provide the foundation for understanding the theory and practice of compilers, learning programming language translation and compiler design concepts. This will help the students to understand language recognition, symbol table management, semantic analysis and code generation.

## Prerequisite

* Formal language and automata theory (BTCS01001)

# Course Outcomes (CO) and mapping with Program Outcomes (PO)

## Course Outcomes

After successful completion of the course, a student will be able to-

1. Comprehend the working of lexical analysis, parsing.

2. Understand the significance of code generation and code optimization.

3. Implement working module of compiler

## CO-PO Mapping

|  | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CO1 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO2 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO3 |  |  |  |  |  |  |  |  |  |  |  |  |

***Green- medium mapping Blue- high mapping***

# Syllabus, Pre-class activity and References

## Teaching and evaluation scheme

| **Teaching Scheme** | | | | **Evaluation Scheme** | |
| --- | --- | --- | --- | --- | --- |
| **Lecture**  **Hours per week** | **Practical**  **Hours per week** | **Tutorial**  **Hours per week** | **Credit** | **Internal Continuous Assessment (ICA)**  **As per Institute Norms**  **(50 Marks)** | **Theory**  **(3 Hrs,**  **100 Marks)** |
| 3 | 2 | 0 | 4 | Marks Scaled to 50 | Marks Scaled to 50 |

## 3.2 Syllabus

| **Unit** | **Description** | **Duration** |
| --- | --- | --- |
| 1. | Introduction: Phases of compilation and overview. Lexical Analysis (scanner): Regular languages, finite automata, regular expressions, regular expression to finite automata, scanner generator (lex, flex). | 07 |
| 2. | Syntax Analysis (Parser): Context-free languages and grammars, push-down automata, LL(1) grammars and top-down parsing, operator grammars, LR(O), SLR(1), LR(1), LALR(1) grammars and bottom-up parsing, ambiguity and LR parsing, LALR(1) parser generator (yacc, bison) | 10 |
| 3. | Semantic Analysis: Attribute grammars, syntax directed definition, evaluation and flow of attribute in a syntax tree. | 05 |
| 4. | Symbol Table: Basic structure, symbol attributes and management. Run-time environment: Procedure activation, parameter passing, value return, memory allocation, scope. | 05 |
| 5. | Intermediate Code Generation: Translation of different language features, different types of intermediate forms. | 04 |
| 6. | Code Improvement (optimization): control-flow, data-flow dependence, local optimization, global optimization, loop optimization, peep-hole optimization. | 05 |
| 7. | Architecture dependent code improvement: instruction scheduling (for pipeline), loop optimization (for cache memory), Register allocation and target code generation. | 05 |
| 8. | Advanced topics: Type systems, data abstraction, compilation of Object Oriented features and non-imperative programming languages | 04 |
|  | **Total hours** | **45** |

## Pre-class activity

Outline for preliminary study to be done for each unit will be provided prior to commencement of each unit. Preliminary study material (video links, presentation, notes etc) will be made available on the student portal. Students are expected to go through this material before attending the upcoming session. It is expected that the students put in at least two hours of self-study for every one hour of classroom teaching. During the lecture session, more emphasis will be given on in-depth topics, practical applications and doubt solving.

## References

| **Text Books:**  1. V. Aho, R. Sethi and J. Ullman “Compilers: Principles, Techniques and Tools”, 2nd Edition, Boston, Massachusetts, USA: Addison-Wesley, 2006.  2. Levine R. John, Tony Mason and Doug Brown “Lex & Yacc”, 2nd Edition, O’Reilly media, 2007. |
| --- |
| **Reference Books:**  1. Bjarne Stroustrup “The Design and Evolution of C++” 2nd edition, Addison-Wesley, 2007. |

***Note: The latest edition of books should be referred.***

# Laboratory details

Students are expected to recall the fundamental theory concepts relevant to the exercise to be performed in the upcoming laboratory.

The following 10 programming exercises will form the submission for laboratory coursework.

| **Sr. No.** | **Week No.#** | **List of Lab Exercises** | **Mapped CO** |
| --- | --- | --- | --- |
| 1 | 1 | Case study on different types of compiler | CO3 |
| 2 | 2 | To check the different keywords and operators (tokens) in the given grammar | CO1 |
| 3 | 3 | To implement Left Recursion | CO1 |
| 4 | 4 | To implement Left factoring | CO1 |
| 5 | 5 | To find the First of the given grammar | CO1 |
| 6 | 6 | To find the Follow of the given grammar | CO1 |
| 7 | 7 | To design a predictive parser for the given grammar | CO1 |
| 8 | 8 | To implement the shift reduce parsers | CO1 |
| 9 | 9 | To implement operator precedence parsers | CO1 |
| 10 | 10 | To implement intermediate code generation | CO2 |

# Tutorial Plan

No Tutorial for this course

# Assessment Policy

## Component wise Continuous Evaluation Internal Continuous Assessment (ICA) and Term End Examination (TEE)

| **Assessment Component** | **ICA (100 Marks)**  **(Marks scaled to 50)** | | | | | **TEE (100 marks)**  **(Marks scaled to 50)** |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Lab Performance** | **Viva** | **Project/ Case study based proposal** | **Class Test 1 and Class Test 2** | **Class Participation** |  |
| **Weightage** | 10% | 5% | 10% | 20% | 5% | 50% |
| **Marks** | 20 | 10 | 20 | 20+20 | 10 | 100 |

## Assessment Policy for Internal Continuous Assessment (ICA)

Assessment of ICA comprises the following components.

1. **Class test 1 and 2 (40 marks)**
   1. Two class tests will be conducted as per the academic calendar.
   2. It may be conducted online/ offline for 20 marks each

1. **Lab performance evaluation (20 marks)**
   1. Lab experiments
      1. Continuous assessment for laboratory experiments will be conducted. There are 10 practicals, each carrying a weightage of 10 marks. At the end of the course, an average of total marks will be taken to obtain marks out of 10.
      2. Discussion of your work with your peers is allowed. However each student is expected to submit his/her original work. Submissions which are very similar will be marked zero. Assessment of the lab work will be carried out based on parameters like timely completion of lab work file, understanding of the experiment performed, originality in the work, involvement of the student, regularity, discipline etc. during the session. There is a 30% penalty on late submission.
   2. One lab is assigned for the recent study of compilers. The students can discuss the projects and research trends of their interest. (Presentations can be included).
2. **Viva (10 marks) -** Lab evaluation and viva will be conducted based on the concepts learnt.
3. **Project based/ Case study based proposal (20 marks)**
   1. Faculty will make group of 3-4 students
   2. Students can work on existing projects/ case study based implementations
   3. Get the topic approval
   4. Implement the paper/ project based on the concepts learnt
   5. Assessment will be based on the idea, the frameworks included, the technical concepts as well as the final report.
4. **Class Participation (10 marks):** The faculty will ask some questions in every class based on the content being taught. The question could be asked to a chosen student. One mark can be given to the correct answer. The idea is to encourage students to pay attention in class and actively participate. These marks will be added in the ICA class participation component.

## Assessment Policy for Term End Examination (TEE)

A written examination of 100 marks for\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ duration will be conducted for the course as per the academic calendar.

# 7. Lesson Plan

| **Session**  **No.** | **Topics** | **Mapped CO** | **Reference** |
| --- | --- | --- | --- |
| 1 | Phases of compilation and overview. | CO1 | TB1, RB1 |
| 2 | Lexical Analysis (scanner): Regular languages | CO1 | TB1, RB1 |
| 3 | Finite automata | CO1 | TB1, RB1 |
| 4 | Regular expressions | CO1 | TB1, RB1 |
| 5 | Regular expression to finite automata | CO1 | TB1, RB1 |
| 6 | Scanner generator (lex, flex). | CO1 | TB1, RB2 |
| 7 | Context-free languages and grammars, push-down automata and bottom-up parsing, | CO1 | TB1, RB1 |
| 8 | LL(1) grammars and top-down parsing | CO1 | TB1, RB1 |
| 9 | LL(1) grammars and top-down parsing | CO1 | TB1, RB1 |
| 10 | LL(1) grammars and top-down parsing | CO1 | TB1, RB1 |
| 11 | Operator grammars | CO1 | TB1, RB1 |
| 12 | Operator grammars | CO1 | TB1, RB1 |
| 13 | LR(O), SLR(1) grammars | CO1 | TB1, RB1 |
| 14 | LR(O), SLR(1) grammars | CO1 | TB1, RB1 |
| 15 | LR(O), SLR(1) grammars | CO1 | TB1, RB1 |
| 16 | Class Test 1 |  |  |
| 17 | LR(1), LALR(1) grammars | CO1 | TB1, RB1 |
| 18 | LR(1), LALR(1) grammars | CO1 | TB1, RB1 |
| 19 | LR(1), LALR(1) grammars | CO1 | TB1, RB1 |
| 20 | LR(1), LALR(1) grammars | CO1 | TB1, RB1 |
| 21 | ambiguity and LR parsing, | CO1 | TB1, RB1 |
| 22 | ambiguity and LR parsing | CO1 | TB1, RB1 |
| 23 | ambiguity and LR parsing | CO1 | TB1, RB1 |
| 24 | LALR(1) parser generator (yacc, bison) | CO1 | TB1, RB1 |
| 25 | Attribute grammars | CO1 | TB1, RB1 |
| 26 | Syntax directed definition, evaluation and flow of attribute in a syntax tree. | CO1 | TB1, RB1 |
| 27 | Basic structure, symbol attributes and management, | CO1 | TB1, RB1 |
| 28 | Run-time environment: Procedure activation | CO1 | TB1, RB1 |
| 29 | Parameter passing, value return | CO1 | TB1, RB1 |
| 30 | memory allocation, scope | CO1 | TB1, RB1 |
| 31 | Translation of different language features | CO1 | TB1, RB1 |
| 32 | Different types of intermediate forms | CO2 | TB1, RB1 |
| 33 | Control-flow, data-flow dependence | CO2 | TB1, RB1 |
| 35 | Class Test 2 |  |  |
| 36 | Control-flow, data-flow dependence graphs | CO2 | TB1, RB1 |
| 37 | Optimization techniques | CO2 | TB1, RB1 |
| 38 | Local optimization, global optimization | CO2 | TB1, RB1 |
| 39 | Loop optimization, peep-hole optimization. | CO2 | TB1, RB1 |
| 40 | Instruction scheduling (for pipeline) | CO2 | TB1, RB1 |
| 41 | loop optimization (for cache memory) | CO2 | TB1, RB1 |
| 42 | Register allocation and target code generation | CO2 | TB1, RB1 |
| 43 | Type systems, data abstraction | CO3 | TB1, RB1 |
| 44 | Compilation of Object Oriented features | CO3 | TB1, TB2 |
| 45 | Non-imperative programming languages | CO3 | TB1,TB2 |

*TB- Text Book, RB - Reference book*

# 8. Teaching-learning methodology

Faculty will make a group of 3-4 students for any group based activity such as class participation, project, presentation etc. Lecture and laboratory session will be conducted as follows-

1. **Lectures:** 
   * Outline for preliminary study to be done for each unit will be provided prior to commencement of each unit.
   * Deeper concepts and applications will be explained through Presentation and Video Lectures.
   * Numerical problems based on concepts will be solved during the session on *smart boards, tablets* or *MS OneNote.*
   * Some practical applications will be simulated in class for better understanding of the concepts, which will be available on *MS Teams* for future reference.
2. **Laboratory:**
   * Lab manual consisting of theory and algorithm to support the lab experiment will be uploaded on MS Teams.
   * Regular lab assessment and grading will be done. Students will be marked based on parameters like completion of lab assignment, originality, logic developed, interaction during the lab, submission, punctuality and discipline

**9. Active learning techniques**

Active learning is a method of learning in which students are actively or experientially involved in the learning process. Following active learning techniques will be adopted for the course.

1. **Muddiest topic:** Faculty will find out the least understood point/topic in the session. This topic is then further explained to ensure that it is understood well.
2. **Blended Learning:** Students will be introduced to the topic at home while the in-depth topics, applications and numerical problems will be discussed by the faculty in the lecture session. Outline for preliminary study to be done for each unit will be provided prior to commencement of each unit. Preliminary study material (video links, presentation, notes etc) will be made available on the MS Teams.
3. **Brainstorming: S**tudents will be asked to generate ideas on a certain topic, category or question while the faculty will facilitate and record the answers on the blackboard/whiteboard.
4. **Case Study / Project based learning:** Students shall be creating groups for learning, discussion and implementation of projects or research. They shall provide a report/ proposal based on the concepts learnt in the course.

**10. Course Material**

Following course material is uploaded on the MS teams

<https://teams.microsoft.com/_#/school/files/Compiler%20Design?threadId=19%3Ac7185174cbca4e6a8e58b11f511b6406%40thread.tacv2&ctx=channel&context=Compiler%2520Design&rootfolder=%252Fsites%252FB.TechCSBSSEMV_A.Y.2021-22%252FShared%2520Documents%252FCompiler%2520Design>

* Course Policy
* Lecture Notes
* Lecture Videos
* Lecture Presentations
* Books / Reference Books / Online links for reference
* Lab Manuals
* List of Program Outcomes

**11. Course Outcome Attainment**

Following means will be used to assess attainment of course learning outcomes.

* Use of formal evaluation components of continuous evaluation, laboratory work, semester end examination
* Informal feedback during course conduction

**12. Academic Integrity Statement**

Students are expected to carry out assigned work under Internal Continuous Assessment (ICA) independently. Copying in any form is not acceptable and will invite strict disciplinary action. Evaluation of corresponding components will be affected proportionately in such cases. Plagiarism detection software will be used to check plagiarism wherever applicable. Academic integrity is expected from students in all components of course assessment.