



**B. P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY**  
**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Lesson plan of CS702 Compiler Design with number of hours/periods, TA/TM, Text/Reference book**

**Text Book:**

**B1.** Alfred V. Aho, Ravi Sethi & Jeffrey. D. Ullman, "Compilers Principles, Techniques & Tools", Pearson Education, third edition, 2007.

**Reference Book:**

**B2.** N Prasad, "Principles of Compiler Design", 2nd Edition, Elsevier

**Web resources**

**W1.** <https://www.univ-orleans.fr/lifo/Members/Mirian.Halfeld/Cours/.../l3-0708-LexA.pdf>

**W2.** [https://www.tutorialspoint.com/compiler\\_design/pdf/compiler\\_design\\_syntax\\_analysis.pdf](https://www.tutorialspoint.com/compiler_design/pdf/compiler_design_syntax_analysis.pdf)

**W3.** <https://web.stanford.edu/class/archive/cs/cs143/.../180%20Semantic%20Analysis.pdf>

**W4:** [www.vssut.ac.in/lecture\\_notes/lecture1422914957.pdf](http://www.vssut.ac.in/lecture_notes/lecture1422914957.pdf)

Lecture No.	Topics to be covered	Ref	Teaching Aids	Testing Method
1	Compilers, Analysis-synthesis model	B1	BB	Discussion
2	The phases of the compiler, Cousins of the compiler.	B1	BB PPT	Discussion Q&A session
3	The role of the lexical analyzer, Tokens, Patterns, Lexemes	B1, W1	BB PPT	Discussion Q&A session
4	Input buffering, Specifications of a token, Recognition of tokens, Finite automata	B1 W1	BB	Discussion Q&A session
5	From a regular expression to DFA	B1, B2 W1	BB	Discussion Q&A session
6	Design of a lexical analyzer generator (Lex).	B2, W1	PPT	Assignments
7	Examples and Problems on NFA to DFA conversion (Gap in the syllabus)	B2, W1	BB, PPT	Problem Solving
8	The role of a parser, Context free grammars, Writing a grammar	B1, W2	PPT	Quiz
9	Top down Parsing, Non-recursive Predictive parsing (LL), ,	B1,B2	BB, PPT	Discussion Q&A session
10	Bottom up parsing	B1, W2	BB, PPT	Quiz
11	Handles, Viable prefixes, Operator precedence parsing,	B1	BB, PPT	Surprise Test
12	LR parsers (SLR, LALR)	B1	BB, PPT	Problem Solving
13	LR parsers (SLR, LALR)	B1, W2	BB, PPT	Problem Solving
14	Parser generators (YACC)	B1	BB,PPT	Assignments
15	Error Recovery strategies for different parsing techniques.	B1	BB,PPT	Quiz

16	Problems on LR Parsers	B2	BB, PPT	Assignments
17	Syntax directed definitions, Construction of syntax trees	B1 W3	BB,PPT BB,PPT	Group Discussion Quiz
18	Bottom-up evaluation of S attributed definitions, L attributed definitions	B1	BB, PPT	Flip Class
19	Bottom-up evaluation of inherited attributes.	B1 W3	BB	Flip Class
20	Problems on Syntax Tree	B2	BB	Assignments
21	Type systems, Specification of a simple type checker,	B1	PPT	Group Discussion Quiz
22	Equivalence of type expressions, Type conversions	B1	BB, PPT	Assignments
23	Source language issues (Activation trees, Control stack, scope of declaration, Binding of names),	B1	BB, PPT	Group Discussion Quiz
24	Storage organization	B1, B2	BB, PPT	Flip Class
25	Intermediate languages, Graphical representation	B1	BB, PPT	Group Discussion Quiz
26	Three-address code, Implementation of three address statements (Quadruples, Triples, Indirect triples).	B1	BB, PPT	Problem Solving
27	Problems on Quadruples, Triples, Indirect triples	B2	BB, PPT	Assignments
28	Introduction, Basic blocks & flow graphs, Transformation of basic blocks, Dag representation of basic blocks,	B1	BB, PPT	Group Discussion Quiz
29	The principle sources of optimization, Loops in flow graph,	B1	BB, PPT	Flip Class
30	Peephole optimization.	1	BB, PPT	Assignments
31	Issues in the design of code generator, a simple code generator	1	BB, PPT	Group Discussion Quiz
32	Register allocation & assignment.	1	BB, PPT	Group Discussion Quiz
33	Revision on Lexical Analysis	1	BB, PPT	Group Discussion Quiz
34	Revision on Parser	1	BB, PPT	Group Discussion Quiz
35	Revision on Intermediate Code Generation	1	BB, PPT	Group Discussion Quiz
36	Revision on Intermediate Code Optimization	1	BB, PPT	Group Discussion Quiz
37	Solutions with Explanations of previous year end semester question paper		BB, PPT	Discussion Q&A session

## SYLLABUS

**Course: Compiler Design**  
**Code: CS702**  
**Contacts: 3L**  
**Credits: 3**

### Module I

#### **Introduction to Compiling [2L]**

Compilers, Analysis-synthesis model , The phases of the compiler, Cousins of the compiler.

#### **Lexical Analysis [5L]**

The role of the lexical analyzer, Tokens, Patterns, Lexemes, Input buffering, Specifications of a token, Recognition of tokens, Finite automata, From a regular expression to an NFA, From a regular expression to NFA, From a regular expression to DFA, Design of a lexical analyzer generator (Lex).

### Module II

#### **Syntax Analysis [8L]**

The role of a parser, Context free grammars, Writing a grammar, Top down Parsing, Non-recursive Predictive parsing (LL), Bottom up parsing, Handles, Viable prefixes, Operator precedence parsing, LR parsers (SLR, LALR), Parser generators (YACC). Error Recovery strategies for different parsing techniques.

#### **Syntax directed translation [4L]**

Syntax directed definitions, Construction of syntax trees, Bottom-up evaluation of S attributed definitions, L attributed definitions, Bottom-up evaluation of inherited attributes.

### Module III

#### **Type checking [3L]**

Type systems, Specification of a simple type checker, Equivalence of type expressions, Type conversions

#### **Run time environments [4L]**

Source language issues (Activation trees, Control stack, scope of declaration, Binding of names), Storage organization (Subdivision of run-time memory, Activation records), Storage allocation strategies, Parameter passing (call by value, call by reference, copy restore, call by name), Symbol tables, dynamic storage allocation techniques.

### Module IV

#### **Intermediate code generation [3L]**

Intermediate languages, Graphical representation, Three-address code, Implementation of three address statements (Quadruples, Triples, Indirect triples).

**Code optimization [4L]**

Introduction, Basic blocks & flow graphs, Transformation of basic blocks, Dag representation of basic blocks, The principle sources of optimization, Loops in flow graph, Peephole optimization.

**Code generations [3L]**


Issues in the design of code generator, a simple code generator, Register allocation & assignment.

**Reference Books:**

1. Aho, Sethi, Ullman - "Compiler Principles, Techniques and Tools" - Pearson Education.
2. Holub - "Compiler Design in C" – PHI
3. Tremblay and Sorenson Compiler Writing-McgrawHill International .
4. Chattopadhyay , S- Compiler Design ( PHI)

**B. P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY**

**Department of Computer Science & Engineering**

<p><b>Course Code</b> : CS 702  <b>Course Name</b> : Compiler Design  <b>Year</b> : 4<sup>th</sup>  <b>Semester</b> : VII  <b>Academic Year</b>: 2017-18</p>	 <p><b>BLOOM's Cognitive Level ref</b></p>
--	--

**COURSE OUTCOMES & PO MAPPING:**

Sl. No.	Description	PO(1..12) mapping	PSO(1..2) mapping	BLOOM's Cognitive Level
CS702.CO1	Recall the finite state automata and interpret, design lexical analyser	PO1, PO2, PO3, PO4	PSO1, PSO2	Apply
CS702.CO2	Explain and Solve problems in Syntax Analysis and Construct different parser	PO1, PO2, PO3, PO4	PSO1, PSO2	Apply
CS702.CO3	Explain and Solve problems in Semantic Analysis using different techniques	PO1, PO2, PO3, PO4	PSO1, PSO2	Analyse
CS702.CO4	Explain code generation schemes.	PO1, PO2, PO3, PO4	PSO1, PSO2	Create
CS702.CO5	Explain optimization of codes and runtime environment	PO1, PO2, PO3, PO4	PSO1, PSO2	Evaluate

PO1	Engineering Knowledge	PO7	Environment & Sustainability	PSO1	Domain Skills 1
PO2	Problem Analysis	PO8	Ethics	PSO2	Domain Skills 2
PO3	Design & Development	PO9	Individual & Team Work		
PO4	Investigations	PO10	Communication Skills		
PO5	Modern Tools	PO11	Project Mgt. & Finance		
PO6	Engineer & Society	PO12	Life Long Learning		

**PSO1:** Students will have proficiency in fundamental engineering and computing techniques and knowledge on contemporary topics like artificial intelligence, data science and distributed computing towards development of optimized algorithmic solutions.

**PSO2:** Students will have capabilities to participate in the development of software and embedded systems through synergized teams to cater to the dynamic needs of the industry and society.

---

Faculty



**PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY**  
**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

ACADEMIC YEAR: 2017-2018 [ODD SEMESTER]

**Odd semester CS702, section=B, Compiler Construction Quiz 1, Full marks 10**

- Q.1 In a Compiler, keywords of a language are recognized during  
(A) parsing of the program (B) the code generation (C) the lexical analysis of the program (D) dataflow analysis
- Q.2 The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?  
A) Finite state automata B) Deterministic pushdown automata C) Non-Deterministic pushdown automata  
D) Turing Machine
- Q.3. The lexical analyzer takes \_\_\_\_\_ as input and produces a stream of \_\_\_\_\_ as output.  
A) Source programs, tokens B) Token, source program C) Either A and B D) None of the above
- Q4, 'Macro' in an 'assembly level program' is  
A) Sub-program B) a complete program C) a hardware portion D) relative coding
- Q5.** Grammar of the programming is checked at \_\_\_\_\_ phase of compiler.  
A) Semantic analysis B) code generation C) syntax analysis D) code optimization
- Q6 . Which of the following is used for grouping of characters into tokens?  
**A) Parser** B) Code optimization C) Code generator D) Lexical analyser
- Q7. A compiler that runs on one machine and produces code for a different machine is called  
A) Cross compilation B) One pass compilation C) Two pass compilation D) None of the above
- Q8. Which of the following is used for grouping of characters into tokens?  
A) Parser B) Code optimization C) Code generator D) Lexical analyser
- Q9. A grammar that produces more than one parse tree for some sentence is called  
A) Ambiguous B) Unambiguous C) Regular D) None of these
- Q10. Intermediate code generation phase gets input from  
A) Lexical analyser B) Syntax analyser C) **Semantic analyser** D) Error handling



**BPPIMT-CSE Dept, July-Dec 2018 CS702, section=B, Compiler Construction Quiz 2, Full marks 10**

- Q.1. Which of the following is the most powerful parsing method?  
A) LL(1)      B) Canonical LR      C) SLR      D) LALR
- Q 2. Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?**  
(A) Removing left recursion alone      (B) Factoring the grammar alone  
(C) Removing left recursion and factoring the grammar      (D) None of these
- Q3. 1(ii) Which one of the following is a top-down parser?**  
(A) Recursive descent parser.      (B) Operator precedence parser.  
(C) An LR(k) parser.      (D) An LALR(k) parser
- Q4. The grammar  $S \rightarrow aSa \mid bS \mid c$  is:**  
A) LL(1) but not LR(1)      B) LR(1) but not LL(1)  
C) Both LL(1) and LR(1)      D) Neither LL(1) nor LR(1)
- Q5. Which of the following is the most powerful parsing method?**  
A) LL(1)      B) Canonical LR      C) SLR      D) LALR
- Q6. Assume that SLR parser for a grammar G has N1 states and the LALR parser for G has N2 states. The relationship between N1 and N2 is:**  
A) N1 is necessarily less than N2.      B) N1 is necessarily equal to N2.  
C) N1 is necessarily greater than N2.      D) None of these.
- Q7. In a bottom-up evaluation of a Syntax directed definition, inherited attributes can**  
A) Always be evaluated.      B) Be evaluated only if the definition is L-attributed.  
C) Be evaluated only if the definition has synthesized attributes.      D) Never be evaluated.
- Q8. YACC builds up**  
A) SLR parsing table      B) Canonical LR parsing table      C) LALR parsing table      D) None
- Q9. We can optimize code by**  
A) Dead code elimination      B) Common subprograms      C) copy intermediate loop      D) loop declaration
- Q10. The optimization technique which is typically applied on loops is**  
A) Removal of invariant computation      B) Peephole optimization      C) Constant folding      D) All of these

ANSWERS

**Q. Which of the following is the most powerful parsing method?**

- A) LL(1)      B) Canonical LR      C) SLR      D) LALR

**Answer: (B) Canonical LR**

**Q5. Grammar of the programming is checked at \_\_\_\_\_ phase of compiler.**

- A) Semantic analysis   B) code generation   C) **syntax analysis**   D) code optimization

**Q3. The lexical analyzer takes \_\_\_\_\_ as input and produces a stream of \_\_\_\_\_ as output.**

- A) **Source programs, tokens**   B) Token, source program   C) Either A and B      D) None of the above

**Q4. 'Macro' in an 'assembly level program' is**

- A) **Sub-program**   B) a complete program   C) a hardware portion      D) relative coding

**Q9. We can optimize code by**

- A) **Dead code elimination**   B) Common subprograms   C) copy intermediate loop   D) loop declaration

**Q. The optimization technique which is typically applied on loops is**

- Removal of invariant computation   B) Peephole optimization   C) Constant folding   **D) All of these**

**Q6 . Which of the following is used for grouping of characters into tokens?**

- A) Parser   B) Code optimization   C) Code generator   **D) Lexical analyser**

**Q6) A compiler that runs on one machine and produces code for a different machine is called**

- A) **Cross compilation**   B) One pass compilation   C) Two pass compilation      D) None of the above

**Q (i). Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?**

(A) Removing left recursion alone      (B) Factoring the grammar alone

(C) Removing left recursion and factoring the grammar      (D) **None of these**

**Answer: (D)**

**Explanation:** Removing left recursion and factoring the grammar do not suffice to convert an arbitrary CFG to LL(1) grammar.

**Q 1(ii) Which one of the following is a top-down parser?**

(A) **Recursive descent parser.**   (B) Operator precedence parser.

(C) An LR(k) parser.      (D) An LALR(k) parser

**Answer: (A)**

**Explanation:** Recursive Descent parsing is LL(1) parsing which is top down parsing.

**Q** Which of the following is used for grouping of characters into tokens?

- A) Parser   B) Code optimization   C) Code generator      **D) Lexical analyser**



Q8. YACC builds up

SLR parsing table B) Canonical LR parsing table C) *LALR parsing table* D) None

Q 1(iii). The grammar  $S \rightarrow aSa \mid bS \mid c$  is:

- |    |                               |    |                        |
|----|-------------------------------|----|------------------------|
| A) | LL(1) but not LR(1)           | B) | LR(1)but not LR(1)     |
| C) | Both LL(1)and LR(1) (correct) | D) | Neither LL(1)nor LR(1) |

Explanation:

First(aSa) = a

First(bS) = b

First(c) = c

All are mutually disjoint i.e no common terminal between them, the given grammar is LL(1).

As the grammar is LL(1) so it will also be LR(1) as LR parsers are more powerful then LL(1) parsers. and all LL(1) grammar are also LR(1)

**So option C is correct.**

Q. Which of the following is the most powerful parsing method?

- A. LL(1)
- a) Canonical LR
- b) SLR
- c) LALR

Answer: (B) Canonical LR

Q. Assume that SLR parser for a grammar G has  $N_1$  states and the LALR parser for G has  $N_2$  states. The relationship between  $N_1$  and  $N_2$  is: [GATE 2003]

- A.  $N_1$  is necessarily less than  $N_2$ .
- B.  $N_1$  is necessarily equal to  $N_2$ .
- C.  $N_1$  is necessarily greater than  $N_2$ .
- D. None of these.

Answer: (B)  $N_1$  is necessarily equal to  $N_2$ .

Q In a bottom-up evaluation of a Syntax directed definition, inherited attributes can

- A. Always be evaluated.
- B. Be evaluated only if the definition is L-attributed.
- C. Be evaluated only if the definition has synthesized attributes.
- D. Never be evaluated.

Answer: (B) Be evaluated only if the definition is L-attributed.

Q . Canonical set of items is given below

$S \rightarrow L > R$

$Q \rightarrow R$ .

On input symbol  $<$  the set has

- (A) a shift-reduce conflict and a reduce-reduce conflict.
- (B) a shift-reduce conflict but not a reduce-reduce conflict.
- (C) a reduce-reduce conflict but not a shift-reduce conflict.
- (D) neither a shift-reduce nor a reduce-reduce conflict.

**Explanation:** The question is asked with respect to the symbol ' $<$ ' which is **not present** in the given canonical set of items. Hence it is neither a shift-reduce conflict nor a reduce-reduce conflict on symbol ' $<$ '. Hence **D** is the correct option.

**Q. Which one of the following is a top-down parser?**

- (A) Recursive descent parser.**
- (B) Operator precedence parser.
- (C) An LR(k) parser.
- (D) An LALR(k) parser

**Explanation:** Recursive Descent parsing is LL(1) parsing which is top down parsing.

**Q. In a compiler, keywords of a language are recognized during**

- (A) parsing of the program
- (B) the code generation
- (C) the lexical analysis of the program**
- (D) dataflow analysis

**3. The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense? (GATE CS 2011)**

- A     Finite state automata (correct)**
- B     Deterministic pushdown automata
- C     Non-Deterministic pushdown automata
- D     Turing Machine

**6. Match all items in Group 1 with correct options from those given in Group 2. (GATE-CS-2009)**

Group 1	Group 2
P. Regular expression	1. Syntax analysis
Q. Pushdown automata	2. Code generation
R. Dataflow analysis	3. Lexical analysis
S. Register allocation	4. Code optimization

- A     P-4, Q-1, R-2, S-3
- B     P-3, Q-1, R-4, S-2 (correct)**
- C     P-3, Q-4, R-1, S-2
- D     P-2, Q-1, R-4, S-3

**1. In a compiler, keywords of a language are recognized during**

- (A) parsing of the program
- (B) the code generation
- (C) the lexical analysis of the program**
- (D) dataflow analysis

**Answer: (C)**

**Explanation:** Lexical analysis is the process of converting a sequence of characters into a sequence of tokens. A token can be a keyword.

2. The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense?

- (A) Finite state automata
- (B) Deterministic pushdown automata
- (C) Non-Deterministic pushdown automata
- (D) Turing Machine

Answer: (A)

3. A canonical set of items is given below

$S \rightarrow L > R$

$Q \rightarrow R$

On input symbol  $<$  the set has

- (A) a shift-reduce conflict and a reduce-reduce conflict.
- (B) a shift-reduce conflict but not a reduce-reduce conflict.
- (C) a reduce-reduce conflict but not a shift-reduce conflict.
- (D) *neither a shift-reduce nor a reduce-reduce conflict.*

Answer: (D)

**Explanation:** The question is asked with respect to the symbol ' $<$ ' which is **not present** in the given canonical set of items. Hence it is neither a shift-reduce conflict nor a reduce-reduce conflict on symbol ' $<$ '.

Hence **D** is the correct option.

4. Which one of the following is a top-down parser?

- (A) *Recursive descent parser.*
- (B) Operator precedence parser.
- (C) An LR(k) parser.
- (D) An LALR(k) parser

Answer: (A)

**Explanation:** Recursive Descent parsing is LL(1) parsing which is top down parsing.

Q 5. The grammar  $S \rightarrow aSa \mid bS \mid c$  is:

- A LL(1) but not LR(1)
- B LR(1) but not LL(1)
- C Both LL(1) and LR(1) (correct)

D Neither LL(1) nor LR(1)

Explanation:

First(aSa) = a

First(bS) = b

First(c) = c

All are mutually disjoint i.e. no common terminal between them, the given grammar is LL(1).

As the grammar is LL(1) so it will also be LR(1) as LR parsers are more powerful than LL(1) parsers. and all LL(1) grammar are also LR(1)  
So option C is correct.

**Q 6. Which of the following suffices to convert an arbitrary CFG to an LL(1) grammar?**

- (A) Removing left recursion alone
- (B) Factoring the grammar alone
- (C) Removing left recursion and factoring the grammar
- (D) None of these

**Answer: (D)**

**Explanation:** Removing left recursion and factoring the grammar do not suffice to convert an arbitrary CFG to LL(1) grammar.

**Q7. Match all items in Group 1 with correct options from those given in Group 2.**

Group 1	Group 2
P. Regular expression	1. Syntax analysis
Q. Pushdown automata	2. Code generation
R. Dataflow analysis	3. Lexical analysis
S. Register allocation	4. Code optimization

- A P-4, Q-1, R-2, S-3
- B P-3, Q-1, R-4, S-2 (correct)
- C P-3, Q-4, R-1, S-2
- D P-2, Q-1, R-4, S-3

**Q8.**

1. Which of the following is the most powerful parsing method?

- A. LL(1)
- a) Canonical LR
- b) SLR
- c) LALR

Answer: (B) Canonical LR

**Q. 9** Assume that SLR parser for a grammar G has N1 states and the LALR parser for G has N2 states. The relationship between N1 and N2 is: [GATE 2003]

- A. N1 is necessarily less than N2.
- B. N1 is necessarily equal to N2.
- C. N1 is necessarily greater than N2.
- D. None of these.

Answer: (B) N1 is necessarily equal to N2.

**Q10.** In a bottom-up evaluation of a Syntax directed definition, inherited attributes can [GATE 2003]

- A. Always be evaluated.
- B. Be evaluated only if the definition is L-attributed.
- C. Be evaluated only if the definition has synthesized attributes.
- D. Never be evaluated.

Answer: (B) Be evaluated only if the definition is L-attributed.

**Q8** Intermediate code generation phase gets input from

Lexical analyser B) Syntax analyser C) *Semantic analyser* D) Error handling