Buddha Institute of Technology

Department of Computer Science & Engineering

Academic Session- 2022-23 (Odd)

LESSON PLAN DETAILS

Semester: V	**Course Code:** KCS-502	**Course:** Compiler Design
Course Faculty: Mr. Abhinandan Tripathi	**Contact Hours /week:** 4+1	**# of credits:** 4
CIE Marks: 50	**SEE Marks:** 100	**Exam Hours:** 3 hrs

Prerequisites if any:

Course Code	Course Name	Topic/s	Semester
KCS-101	Programming for Problem Solving	Operators, Decision control statement, looping & Translators	I
KCS-301	Data Structures	Stack	III
KCS-303	Theory of Automata & Formal Languages	Grammar & its type, Parse Tree	IV

Content delivery by using	Chalk and Board, PPT and Video Lectures

COURSE SYLLABUS:

Unit #	Contents of Module	Hrs	COs
1	**Introduction to Compiler:** Phases and passes, Bootstrapping, Finite state machines and regular expressions and their applications to lexical analysis, Optimization of DFA-Based Pattern Matchers implementation of lexical analyzers, lexical-analyzer generator, LEX compiler, Formal grammars and their application to syntax analysis, BNF notation, ambiguity, YACC. The syntactic specification of programming languages: Context free grammars, derivation and parse trees, capabilities of CFG	**12**	**CO1**
2	**Basic Parsing Techniques:** Parsers, Shift reduce parsing, operator precedence parsing, top down parsing, predictive parsers Automatic Construction of efficient Parsers: LR parsers, the canonical Collection of LR (0) items, constructing SLR parsing tables, constructing Canonical LR parsing tables, Constructing LALR parsing tables, using ambiguous grammars, an automatic parser generator, implementation of LR parsing tables.	**24**	**CO2**
3	**Syntax-directed Translation:** Syntax-directed Translation schemes, Implementation of Syntax-directed Translators, Intermediate code, postfix notation, Parse trees & syntax trees, three address code, quadruple & triples, translation of assignment statements, Boolean expressions, statements that alter the flow of control, postfix	**13**	**CO3**

	translation, translation with a top down parser. More about translation: Array references in arithmetic expressions, procedures call, declarations and case statements.		
4	**Symbol Tables:** Data structure for symbols tables, representing scope information. Run Time Administration: Implementation of simple stack allocation scheme, storage allocation in block structured language. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors semantic errors.	**10**	**CO4**
5	**Code Generation:** Design Issues, the Target Language. Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, Code Generator. Code optimization: Machine-Independent Optimizations, Loop optimization, DAG representation of basic blocks, value numbers and algebraic laws, Global Data-Flow analysis.	**12**	**CO5**

Course Outcomes:

At the end of the course, the students will be able to:

CO1	**KCS-502.1**	Summarize the concept of different phases and passes of compiler.	L2
CO2	**KCS-502.2**	Construct the parse tree using different parsing techniques.	L3
CO3	**KCS-502.3**	Utilize the concept of SDT, intermediate code generation & three address codes.	L3
CO4	**KCS-502.4**	Classify the concept of symbol table, storage allocation & error detection & recovery.	L2
CO5	**KCS-502.5**	Apply the different code optimization techniques.	L3

Mapping of COs v/s POs:

COs	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
KCS-502.1	2	1	-	-	1	-	-	-	-	-	-	1
KCS-502.2	2	2	2	2	1	-	-	-	-	-	-	2
KCS-502.3	2	2	1	2	1	-	-	-	-	-	-	2
KCS-502.4	2	2	1	2	1	-	-	-	-	-	-	2
KCS-502.5	2	1	-	-	1	-	-	-	-	-	-	2
Average	2.00	1.60	1.33	2.00	1.00	-	-	-	-	-	-	1.80

Lesson Plan Details

Mapping of COs v/s PSOs:

COs	PSO1	PSO2	PSO3
KCS-502.1	1	1	-
KCS-502.2	1	2	-
KCS-502.3	1	2	-
KCS-502.4	1	1	-
KCS-502.5	1	1	-
Average	1	1.4	-

Correlation levels: 1-Slight (Low) 2-Moderate (Medium) 3-Substantial (High)

Topics to be covered beyond syllabus	Peephole Optimization

Assessment Methodologies:

Sl. No.	Description	Type
1	Internal assessment (Tutorials/Assignments & Class Tests)	Direct
2	University exam	Direct
3	Student feedback	Indirect

Lesson Plan Details

LESSON PLAN

Lecture #	Unit #	Topics	RBT Levels	CO Mapping	Scheduled Date	Conducted Date	References	Remarks	Faculty Sign
1	I	Introduction to Compiler, Phases and passes	**L2**	**CO1**	22-08-22		T1, Ch-1		
2	I	Phases and passes (cont.)	**L2**		23-08-22		T1, Ch-1		
3	I	Bootstrapping, Finite state machines	**L2**		24-08-22		T1, Ch-1		
4	I	Regular expressions and their applications to lexical analysis	**L2**		25-08-22		T1, Ch-1		
5	I	Optimization of DFA-Based Pattern Matchers implementation of lexical analyzers	**L2**		27-08-22		T1, Ch-3		
6	I	**Tutorial-1**			29-08-22				
7	I	Lexical-analyzer generator, LEX-compiler	**L2**		30-08-22		T1, Ch-3		
8	I	Formal grammars and their application to syntax analysis , BNF notation	**L2**		31-08-22		T1, Ch-3		
9	I	YACC, The syntactic specification of programming languages: Context free grammars	**L2**		01-09-22		T1, Ch-3		
10	I	Derivation and Parse trees,	**L2**		03-09-22		T1, Ch-3		
11	I	Ambiguity, capabilities of CFG	**L2**		05-09-22		T1, Ch-3		
12	I	**Tutorial-2**			06-09-22				
13	II	Basic Parsing Techniques: Parsers	**L3**		07-09-22		T1, Ch-4		
14	II	Top down parsing	**L3**		08-09-22		T1, Ch-4		
15	II	Predictive parsers	**L3**		10-09-22		T1, Ch-4		
16	II	Predictive parsers (Cont.)	**L3**		12-09-22		T1, Ch-4		
17	II	**Tutorial-3**			13-09-22		T1, Ch-4		
18	II	Predictive parsers (Cont.)	**L3**		14-09-22		T1, Ch-4		

19	II	Shift reduce parsing	**L3**	**CO2**	15-09-22		T1, Ch-4		
20	II	Shift reduce parsing (Cont.)	**L3**		17-09-22		T1, Ch-4		
21	II	Operator precedence parsing	**L3**		22-09-22		T1, Ch-4		
22		**Revision-1**			24-09-22				
23	II	**Tutorial-4**			26-09-22				
24	II	Operator precedence parsing (Cont.)	**L3**		27-09-22		T1, Ch-4		
25	II	Automatic Construction of efficient Parsers: LR parsers	**L3**		28-09-22		T1, Ch-4		
26	II	The canonical Collection of LR(0) Items	**L3**		29-09-22		T1, Ch-4		
27		**Revision-2**			01-10-22				
28	II	Constructing SLR parsing tables	**L3**		03-10-22		T1, Ch-4		
29	II	**Tutorial-5**			06-10-22		T1, Ch-4		
30		**Revision-3**			08-10-22				
31	II	Constructing Canonical LR items	**L3**		10-10-22		T1, Ch-4		
32	II	Constructing Canonical LR parsing Tables	**L3**		11-10-22		T1, Ch-4		
33	II	Constructing LALR parsing tables using ambiguous grammars	**L3**		12-10-22		T1, Ch-4		
34	II	An automatic parser generator, implementation of LR parsing Tables	**L3**		13-10-22		T1, Ch-4, WR2		
35		**Revision-4**			15-10-22				
36	II	**Tutorial-6**			17-10-22				
37	III	Syntax-directed Translation: Syntax-directed Translation scheme	**L2**		18-10-22		T1, Ch-5, T2, Ch-4		
38	III	Implementation of Syntax directed Translators	**L2**		19-10-22		T1, Ch-5		

Lesson Plan Details

39	III	Intermediate code: three address code, quadruple & triples	L3	CO3	20-10-22		T1, Ch-6, T2, Ch-5		
40		**Revision-5**			22-10-22				
41	III	Postfix notation, Parse trees	L3		27-10-22		T1, Ch-6		
42		**Revision-6**			29-10-22				
43	III	**Tutorial-7**			31-10-22				
44		**Revision-7**			05-11-22				
45	III	Syntax trees, Translation of assignment statements	L3		07-11-22		T1, Ch-6, WR1		
46	III	Boolean Expressions, Statements that alter the flow of control, Postfix translation, Translation with a top down parser	L3		08-11-22		T1, Ch-6, WR1		
47	III	More about translation: Array references in arithmetic expressions	L3		14-11-22		T1, Ch-6		
48	III	Procedures call, declarations, Case statements	L3		15-11-22		T1, Ch-6		
49	III	**Tutorial-8**			16-11-22				
50	IV	Symbol Tables: Data structure for symbols tables	L2	CO4	17-11-22		T1, Ch-7		
51		**Revision-8**			19-11-22				
52	IV	Representing scope information	L2		21-11-22		T1, Ch-7		
53	IV	Run-Time Administration	L2		22-11-22		T1, Ch-7		
54	IV	Implementation of simple stack allocation scheme	L2		23-11-22		T1, Ch-7		
55	IV	Storage allocation in block structured language	L2		24-11-22		T1, Ch-7		
56		**Revision-9**			26-11-22				
57	IV	Error Detection & Recovery: Lexical Phase errors	L2		28-11-22		T1, Ch-7, WR3		

Lesson Plan Details

58	IV	Syntactic phase errors, Semantic Errors	**L2**		29-11-22		T1, Ch-7		
59	IV	**Tutorial-9**			30-11-22				
60	V	Code Generation: Design Issues The Target Language, Addresses in the Target Code	**L2**	**CO5**	01-12-22		T1, Ch-8, WR3		
61		**Revision-10**			03-12-22				
62	V	Basic Blocks and Flow Graphs	**L2**		05-12-22		T1, Ch-8		
63	V	Optimization of Basic Blocks	**L3**		06-12-22		T1, Ch-8		
64	V	Optimization of Basic Blocks (Cont.)	**L3**		07-12-22		T1, Ch-8		
65	V	Code Generator	**L3**		08-12-22		T1, Ch-8		
66		**Revision-11**			10-12-22				
67	V	Code optimization: Machine-Independent Optimizations ,Loop optimization	**L3**		12-12-22		T1, Ch-8		
68	V	DAG representation of basic blocks, value numbers and algebraic laws, Global Data-Flow analysis	**L3**		13-12-22		T1, Ch-8		
69	V	**Peephole Optimization**	**L3**		14-12-22		T1, Ch-8		
70	V	**Tutorial-10**			15-12-22				
71		**Revision-12**			17-12-22				

*L1 – Remembering; L2 – Understanding; L3 – Applying; L4 – Analysing; L5 – Evaluating; L6 - Creating

Syllabus for Class Tests:

Class Tests	**Planned Syllabus**	**Actual Syllabus**
CT-1	Lecture 1 to 20	
CT-2	Lecture 21 to 43	
Pre-AKTU	Full Syllabus	

Text Books:

Lesson Plan Details

T1. Aho, Sethi & Ullman, "Compilers: Principles, Techniques and Tools", Pearson Education.
T2. V Raghvan, "Principles of Compiler Design", TMH.

Web References:
WR1. https://www.geeksforgeeks.org/introduction-of-compiler-design/
WR2. https://www.javatpoint.com/compiler-tutorial
WR3. https://www.guru99.com/compiler-design-tutorial.html

Sample Questions:

Q. #	Questions	RBT Level	COs
	UNIT-1		
1	What is the difference between parse tree and abstract syntax tree? **[AKTU2021-22]**	L2	CO1
2	Define the terms Language Translator and compiler. **[AKTU2021-22]**	L2	CO1
3	What is meant by viable prefixes? **[AKTU2018-19]**	L2	CO1
4	What are the classifications of compiler? **[AKTU2018-19]**	L2	CO1
5	What are the two parts of compilation? Explain briefly. **[AKTU2018-19]**	L2	CO1
6	What is YACC? Discuss it. **[AKTU2020-21]**	L2	CO1
7	What are the compiler design tools?	L2	CO1
8	Describe the Front End Of A Compiler?	L2	CO1
9	What tools are used for compiler construction?	L2	CO1
10	What is an overview of the structure of a typical compiler?	L2	CO1
	UNIT-2		
1	Remove the left recursion: **[AKTU2015-16]** S→A|B A→Aa|€ B→b|Sc	L3	CO2
2	Remove left factoring: E→ E+T|T , T→int|(E)	L3	CO2
3	Compute the operator precedence relation for the grammar: S→ a|^|(T) T→T,s|S Is it an operator precedence grammar?	L3	CO2
4	Compute the FIRST and FOLLOW of the following grammar: **[AKTU2018-19]** **(a)** A→BC B→Ax|x C→yC|y **(b)** S→aS|Ab A→XYZ|€ X→cS|€ Y→dS|€ Z→eS **(c)** S→XS|dS|€ X→Y|Zb|aY Y→cZ Z→e **(d)** S→Aa|Bb|cC C→Ab|Ba A→D B→D D→€	L3	CO2
5	S→AA A→aA|b **(i)** Show that the grammar is LR(1) or not **(ii)** Show that the grammar is LALR or not?	L3	CO2

Lesson Plan Details

6	Construct the LL(1) parsing table for this grammar. Is grammar LL(1)? E→FT T→+FT|€ F→ (E)|id Parse the string id++id id, this string may or may not be legal string.	L3	CO2
7	Consider the following grammar : S→ (L)|a L→L,S|S **(a)** Construct operator precedence relation table. **(b)** Find operator precedence function table.	L3	CO2
8	Check whether the following grammar is SLR (1) or not. Explain your answer with Reasons. S→ L=R S→R L→*R L→id R→L	L3	CO2
9	Construct an LALR Parsing table for the following grammar: **[AKTU2021-22]** S →BB B→ aB / b	L3	CO2
10	Construct the predictive parser for the following grammar **[AKTU2021-22]** E-> E+T |T T-> T*F|F F->id	L3	CO2
UNIT-3			
1	Differentiate between syntax & semantic analysis.	L2	CO3
2	Find the three address code for : **(a)** w=a*(b+c)/(d+e) **(b)** –a+b/c↑d↑e*f/g	L3	CO3

3	Write the three address code (TAC) for the following code segment:
	C=0;
	do{
	if(a<b) then X++;else X-
	-;
	C++;
	} while(C<S)
		L3	CO3

4	Write the difference between S-attribute & L-attribute definition.	L2	CO3
5	Write the quadruple, triple, indirect triple for the following expression (x + y)*(y + z) + (x + y + z) **[AKTU2021-22]**	L3	CO3
6	Write syntax directed definition for a given assignment statement: S → id=E E → E+E E →E*E E→-E E → (E) E→id **[AKTU2021-22]**	L3	CO3
7	Write a note on the specification of a simple type checker	L2	CO3
8	Explain intermediate code representations?	L2	CO3

Lesson Plan Details

9	Construct a quadruple, triples for the following expression: a +a*(b-c)+(b-c)*d	L3	CO3
10	What is a three address code? Mention its types. How would you implement the three address statements? Explain with examples.	L2	CO3
	UNIT-4		
1	What are the differences between lexical, syntactic & semantic errors? **[AKTU2017-18]**	L2	CO4
2	When dangling references occurs? **[AKTU2018-19]**	L2	CO4
3	Write the concepts of call by value & call by reference with suitable example.	L2	CO4
4	Give an example to show how DAG is used for register allocation. **[AKTU2021-22]**	L2	CO4
5	Explain various storage allocation strategies with examples.	L2	CO4
6	Define Symbol Table. **[AKTU2020-21]**	L2	CO4
7	Explain the detection & recovery method of syntactic error with suitable example.	L2	CO4
8	Explain the issue and the difference between the heap allocated activation records versus stack allocated activation records.	L2	CO4
	UNIT-5		
1	Define global dataflow analysis.	L2	CO5
2	Define constant folding & reduction in strength?	L2	CO5
3	Explain the following with an example: a) Redundant sub expression elimination b) Frequency reduction c) Copy propagation	L2	CO5

4	Discuss how induction variables can be detected and eliminated from the given intermediate code **[AKTU2021-22]**
	B2: i:= i+1
	t1:=4*j
	t2:=a[t1]
	if t2<10 goto B2
		L3	CO5

5	Discuss the following terms: **[AKTU2021-22]** i. Basic block ii. Next use information iii. Flow graph	L2	CO5

6	Construct the flow graph for the following code segment:
	fact(n)
	{
	int f=1;
	for (i=2; i ≤ n; i++)
	f=f*I;
	return f;
	}
	[AKTU2020-21]
		L3	CO5

Lesson Plan Details

7	What is induction variable elimination?	L2	CO5
8	Explain in detail about peephole optimization. **[AKTU2018-19]**	L2	CO5
9	Explain in detail the procedure that eliminates global common sub expression	L2	CO5

Course Faculty **HOD**

Lesson Plan Details