COMPILER DESIGN (ECS-306)

Teacher Name:

Ms. Ankita Gautam

Course Structure

Sr. No.	Course Code	Course Name	Credits	Details of Sessional Marks				ESM	Total Marks
				CT	TA	Lab	Total		
1.	ICS-603	Compiler Design	4 (3-1-0)	30	20	-	50	50	100

Prerequisite: Theory of Automata and Formal Languages (ECS-305)

Course Content:

Unit-1:

Introduction to Compiler, Phases and passes, Bootstrapping, Finite automata & regular expressions and their applications to lexical analysis, Implementation of lexical analyzers, lexical-analyzer generator, LEX-compiler, The syntactic specification of Programming languages: Context free grammars, derivation and parse trees, capabilities of CFG, Application of grammars in syntax analysis, ambiguity and BNF notation, YACC.

Unit-2:

Basic Parsing Techniques: Parsers, top down parsing, Shift reduces parsing, operator precedence 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, constructing LALR sets of items.

Unit-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 translation, translation with a top down

parser. More about translation: Array references in arithmetic expressions, procedures call, declarations, Case statements.

Unit-4:

Symbol Tables: Data structure and 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.

Unit-5:

Introduction to code optimization: Loop optimization, the DAG representation of basic blocks, value numbers and algebraic laws, Global Data-Flow analysis.

Text and References Books:

1. Aho, Sethi & Ullman, "Compiler Design", Addition Wesley.
2. Kenneth C. Louden, “Compiler Construction: Principles and Practice”, Thomson Brooks Publication.
3. Allen I. Holub, “Compiler Design in C”, PHI Publications.

Course Outcomes:

1. Describe the role of each phase of a compiler with its construction tools. (Understand)
2. Develop a Lexical Analyzer for recognizing tokens of a given language with an understanding of symbol table management and error handling. (Apply)
3. Construct top-down, bottom-up, operator precedence and SLR parsers with an understanding of Context Free Grammars and syntax analysis. (Apply)
4. Design and develop semantic analyzers for type-checking and intermediate code generators to translate the source program into an intermediate code. (Apply)
5. Construct code optimizers to optimize the target code generated. (Apply)

Lesson Plan

LECTURE PLAN

COMPILER DESIGN (ICS-603)

Lecture No.	Unit/Topic/Subtopic	Teaching Method(s)	Learning Material	Remarks if any
	UNIT 1:			
1.	Introduction to Compiler, Phases and passes, Bootstrapping	Chalk and Board	Books and Notes	
2.	Finite automata and regular expressions and their applications to	Chalk and Board	Books and Notes	

	lexical analysis,			
3.	Implementation of lexical analyzers, lexical-analyzer generator, LEX compiler.	Chalk and Board	Books and Notes	
4.	The syntactic specification of Programming languages: Context free grammars,	Chalk and Board	Books and Notes	
5.	derivation and parse trees, capabilities of CFG.	Chalk and Board	Books and Notes	
6.	Application of grammars in syntax analysis,	Chalk and Board	Books and Notes	
7.	ambiguity and BNF notation, YACC.	Chalk and Board	Books and Notes	
	UNIT – 2			
8.	Basic Parsing Techniques: Parsers, top down parsing.	Chalk and Board	Books and Notes	
9.	Shift reduces parsing	Chalk and Board	Books and Notes	
10.	operator precedence parsing	Chalk and Board	Books and Notes	
11.	predictive parsers.	Chalk and Board	Books and Notes	
12.	Automatic Construction of efficient Parsers: LR parsers,	Chalk and Board	Books and Notes	
13.	the canonical Collection of LR(0) items	Chalk and Board	Books and Notes	
14.	the canonical Collection of LR(0) items	Chalk and Board	Books and Notes	
15.	constructing SLR parsing tables,	Chalk and Board	Books and Notes	
16.	constructing SLR parsing tables,	Chalk and Board	Books and Notes	
17.	constructing Canonical LR parsing tables,	Chalk and Board	Books and Notes	
18.	constructing Canonical LR parsing tables,	Chalk and Board	Books and Notes	
19.	Constructing LALR parsing tables, using ambiguous grammars,	Chalk and Board	Books and Notes	
20.	Constructing LALR parsing tables, using ambiguous grammars,	Chalk and Board	Books and Notes	
21.	An automatic parser generator,	Chalk and	Books	

	implementation of LR parsing tables,	Board	and Notes	
22.	constructing LALR sets of items.	Chalk and Board	Books and Notes	
	UNIT- 3			
23.	Syntax-directed Translation schemes, Implementation of Syntax directed Translators,	Ppt	Books and Notes	
24.	Intermediate code, postfix notation,	Ppt	Books and Notes	
25.	Parse trees & syntax trees,	Ppt	Books and Notes	
26.	Three address code, quadruple & triples,	Ppt	Books and Notes	
27.	Translation of assignment statements, Boolean expressions, statements that alter the flow of control,	Ppt	Books and Notes	
28.	Postfix translation, translation with a top down parser postfix translation	Ppt	Books and Notes	
29.	translation with a top down parser	Ppt	Books and Notes	
30.	More about translation: Array references in arithmetic expressions,	Ppt	Books and Notes	
31.	Procedures call, declarations, case statements.	Ppt	Books and Notes	
	UNIT-4 :			
32.	Symbol Tables: Data structure and representing scope information.	Ppt	Books and Notes	
33.	Run-Time Administration: Implementation of simple stack allocation scheme	Ppt	Books and Notes	
34.	storage allocation in block structured language	Ppt	Books and Notes	
35.	Error Detection & Recovery: Lexical Phase errors,	Ppt	Books and Notes	
36.	Syntactic phase errors semantic errors.	Ppt	Books and Notes	
	UNIT-5 :			
37.	Introduction to code optimization.	Ppt	Books and Notes	
38.	Loop optimization	Ppt	Books and Notes	
39.	The DAG representation of basic blocks	Ppt	Books and Notes	
40.	Value numbers and algebraic laws, Global Data-Flow analysis.	Ppt	Books and Notes	

Assignments

Assignment 1
Instructor: Ankita Gautam
Compiler Design (2018-19)

- Assignment should be handwritten.
- Don't Copy
- Deadline: 13/02/2019 (5.00 Pm)

1. Write short note on bootstrapping.
2. Define the terms Language Translator and compiler.
3. Write Regular Expression for specifying Identifiers and constant of C.
4. Define LEX. Explain the use and form of lex program with an example.
5. Explain the need for dividing the compilation process into various phases and explain its functions.
6. Define Regular Expressions and Regular Grammar.
7. How to remove Left Factoring and Left recursion in a grammar.
8. What is ambiguous grammar? Explain with the help of an example.
9. What are the applications of Finite automata and regular expression in Lexical analysis?
10. Calculate first and follow for the following grammar?
S->xABC
A->a|bbD
B->a|ε
C->b| ε D->c| ε

11. Construct the recursive decent parser for the string id*(id+id) following grammar?
E-> E+T/T
T-> T*F/F
F->(E)/id

12. Consider the grammar
S->xABC
A->a|bbD
B->a|ε
C->b| ε
D->c| ε
Derive the predictive parsing table.

Assignment 2
Instructor: Ankita Gautam
Compiler Design (2018-19)

- Assignment should be handwritten.
- Don't Copy
- Deadline: 13/03/2019 (5.00 Pm)
- Give explanation for all the MCQs

1. For a given grammar if SLR(1) has n1 states, LALR(1) has n2 states, LR(1) has n3 states which of the following is true?
    a. n2=n3
    b. n2<=n3
    c. n2<n3
    d. none

2. Consider the grammar given below:
    A → SB | S
    B →; S B | ; S
    S → a

    Convert it into operator grammar.

3. Consider SLR(1) and LALR(1) tables for CFG. Which of the following is false?
    a. Goto of both tables may be different
    b. Shift entries are identical in both tables
    c. Reduce entries in tables may be different
    d. Error entries in tables may be different
4. Consider the grammar S → CC, C → cC | d is
    a. LL(1)
    b. SLR(1) but not LL(1)
    c. LALR(1) but not SLR(1)
    d. LR(1) but not LALR(1)

5. Which of the following is the most powerful parsing method?
    a. LL(1)
    b. CLR(1)
    c. SLR(1)
    d. LALR(1)

6. The following grammar is [ ]
    S → ABC
    A → 0A1 | ε
    B → 1B | ε
    C → 1C0 | ε
    a. LL(1)
    b. LR(0)
    c. not LL(1)
    d. not LL(1) and not LR(0)

7. Check whether the following grammar is LR(0), SLR(1).
    E → bEa | aEb | ba

8. Check if the following grammar is SLR(1), CLR(1).

    S → L = R | R

L → * R | id
R → L

9. How many conflicts occur in DFA with LR(1) items for the following grammar?

S → SS | a | c

10. Consider the following two sets of LR(1) items of an LR(1) grammar.

X -> c.X, c/d

X -> .cX, c/d

X -> .d, c/d

X -> c.X, $

X -> .cX, $

X -> .d, $

1. Cannot be merged since look aheads are different.
2. Can be merged but will result in S-R conflict.
3. Can be merged but will result in R-R conflict.
4. Cannot be merged since goto on c will lead to two different sets.

a. 1 only

b. 2 only

c. 1 and 4 only

d. 1, 2, 3, and 4

Assignment 3

1. Explain syntax directed definition.
2. Explain the Translation scheme of SDD.

3. Draw the syntax tree and DAG for the following expression:

(a*b)+(c-d)*(a*b)+b

4. Differentiate between synthesized translation and inherited translation.
5. What is intermediate code and write the two benefits of intermediate code generation.
6. Write short notes on:
    a. Abstract syntax tree
    b. Polish notation
7. Write quadruples, triples and indirect triples for the expression:

    -(a*b)+(c+d)-(a+b+c+d)

Assignment 4

1. Define Symbol table. Explain different types of Data structure for symbol table
2. Draw the format of Activation Record in stack allocation and explain each field in it.
3. Distinguish between static scope and dynamic scope. Briefly explain access to non-local names in static scope.
4. What is meant by data flow equation?
5. What is activation record? Write the various fields of Activation Record.
6. What are the functions of error handler?
7. Write a short note on Error Detection and Recovery.
8. Classify the errors and discuss the errors in each phase of Compiler.

Assignment 5

1. Construct the DAG for the following basic blocks

    a. t1:=4*i
    b. t2:=a[t1]
    c. t3:=4*i
    d. t4:=b[t3]
    e. t5:=t2*t4
    f. t6:=prod+t5
    g. prod:=t6
    h. t7:=i+1
    i. i:=t7
    j. if i<=20 goto 1
2. Explain the simple code generator and generate target code sequence for the following statement d:=(a-b)+(a-c)+(a-c)
3. What is the role of peephole optimization in compilation process.
4. Consider the following C code segment.
    for (i = 0, i<n; i++)
    {

```
  for (j=0; j<n; j++)
  {
    if (i%2)
    {
      x += (4*j + 5*i);
      y += (7 + 4*j);
    }
  }
}
```

Which one of the following is false?
A: The code contains loop invariant computation
B: There is scope of common sub-expression elimination in this code
C: There is scope of strength reduction in this code
D: There is scope of dead code elimination in this code

5. Consider the following code segment.

```
x = u - t;
y = x * v;
x = y + w;
y = t - z;
y = x * y;
```

The minimum number of total variables required to convert the above code segment to static single assignment form is:

6. What is control and data flow analysis? Explain with example.

7. Write a short note on:

a. Flow graph (with example)
b. Dominators
c. Natural loops
d. Inner loops
e. Reducible flow graphs