Exam. Code: 0918
Sub. Code: 6410

2054

B.E. (Computer Science and Engineering)
Sixth Semester
CS-604: Complier Design

Time allowed: 3 Hours **Max. Marks: 50**

NOTE: ***Attempt five questions in all, including Question No. 1 (Section-A) which is compulsory and selecting two questions each from Section B-C.***

x-x-x

	Section-A	
Q1.	a) Differentiate between compiler and assembler. b) What is the role of input buffering in lexical analyzer? c) Define left factoring with example. d) How can conflicts occur in LALR parser? e) Why loop optimization is important than other code optimization?	10
	Section-B	
Q2.	a) Define compiler. Explain the various phases of compiler in detail with neat sketch. Also compare one pass and two pass compilers.	5
	b) Define symbol table. Discuss any two data structures suitable for it and compare the merits and demerits.	5
Q3.	a) A lexical analyzer uses the following patterns to recognize three tokens T1, T2, and T3 over the alphabet {a,b,c}. T1: a?(b|c)*a T2: b?(a|c)*b T3: c?(b|a)*c If the string **bbaacaba** is processes by the analyzer, Identify all correct combinations of (T1, T2, T3) that the lexer can utilize for recognizing the string. Then select the best possible match out of this. Justify the reason for your decision.	6
	b) Write Lex program to extract email addresses from a given corpus of text.	4
Q4.	a) Construct an LALR parsing table for the given context-free grammar – S->aAd| bBd| aBe| bAe A->c B->c	7
	b) Write short note on YACC.	3
	Section-C	
Q5.	D ->TL T -> int T -> real T-> double L -> L, id L -> id For the above grammar, write suitable semantic rules for extracting type information and store that information into symbol table. Explain how a bottom-up parser can actually implement such a scheme. Specify the implementations mechanism. Explain with suitable example with string **double a, b, c**	10

P.T.O.

Q6.	a) Build the control flow graph for the given code. Evaluate the number of nodes and edges, loops in the control flow graph constructed. b) Explain the graph-based scheme for register assignment and allocation.	1. i = 1 2. j = 1 3. t1 = 5 * i 4. t2 = t1 + j 5. t3 = 4 * t2 6. t4 = t3 7. a[t4] = –1 8. j = j + 1 9. if j <= 5 goto(3) 10. i = i + 1 11. if i < 5 goto(2) 12. else goto(8) 13. exit	5 5
Q7.	a) Construct the DAG for the following basic block: d= b* c e=a +b b=b*c a=e-d b) Simplify the three-address code assuming 1. Only a is live on exit from the block. 2. a, b, and c are live on exit from the block. c) Use the code generator algorithm to generate code for the quadruples.		3 3 4

x-x-x