Compiler Design

Question Bank

UNIT 1

1. Discuss all the phases of compiler with a diagram.
2. Classify approach would you use to recover the errors in lexical analysis phase?
3. Describe the interactions between the lexical analyzer and the parser.
4. Summarize in detail about how the tokens are specified by the compiler with suitable example.
5. Define Lex and Lex specifications. How lexical analyzer is constructed using lex? Give an example.Analyze and identify the symbol table for the following statements int a,b; float c; char z;
6. Design a FA from given regular expression 10 + (0 + 11)0* 1 **or** regular expression.((ε /a)b*)*
7. Solve the given regular expression into NFA using Thompson construction
8. (a/b)* abb (a/b)*.

ii) ab*/ab

1. Write a short note on:
2. YACC
3. Pass
4. Bootstrapping
5. LEX Compiler
6. Tokens, Patterns and Lexemes

UNIT 2

1. Differentiate Top Down parsing and Bottom Up parsing? (Short answers)
2. Define Recursive Descent Parsing. (Short answers)
3. Compute FIRST and FOLLOW for the following grammar. (Short answers)

S → AS

S →b

A →SA

A → a

1. Explain synthesized and inherited atributes with example. (Short answers)
2. Describe on detail about the various types of parser.
3. (i)Consider the following grammar

S →AS|b

A→SA|a.

Construct the SLR parse table for the grammar.

(ii)Show the actions of the parser for the input string “abab”.

1. (i) Show SLR parsing table for the following grammar

A->(A)|a

(ii)Differentiate SLR and CLR

1. (i)Give the predictive parser table for the following grammar.

S→ (L) | a

L→ L, S | S

(ii)Parse the string (a, (a, a)) .

1. Analyze the give grammar to construct predictive parser

S→ +SS | *SS | a with the string “+*aa.

1. Explain left recursion and Left Factoring.Eliminate left recursion and left factoring for the following grammar.

E → E + T | E - T | T

T → a | b | (E).

1. Evalute and draw a annotated parsing tree for given SDT arithmatic expression 4-6/3+5. Given grammar

E > E_1+T, E > E_1-T, E > T, T > T_1 * F, T > T_1/F, T > F, F > E, F > num.

Define Canonical Collection of LR(0) items and draw a DFA and table of given grammar.

S → AA

A → aA | b

1. Explain **LALR(1)** parsing and draw a parsing table of given **LALR (1)** Grammar
2. S → AA
3. A → aA
4. A → b

Consider the following grammar :

S → 1 AB / ∈

A → 1 AC / OC

B → OS

C → 1

and test that whether the grammar is LL(1) or not

UNIT 3

1. Compare synthesized attributes and inherited attributes. (Short Answer)
2. What is Annotated parse tree? (Short Answer)
3. What is a syntax tree? Draw the syntax tree for the assignment statement a := b * -c + b * -c. (Short Answer)
4. Illustrate the methods of implementing three-address statements. (Short Answer)
5. Test whether the following rules are L-attribute or not? Semantic rules. (Short Answer)

A.s = B.b;

B.i = f(C.c,A.s)

1. Create postfix notation for the given expression a+b*c. (Short Answer)
2. Show the three address code sequence for the assignment statement. d=(a-b)+(a-c)+(a-c). (Short Answer)
3. Compare three address code for expression with the Incremental translation.
4. What is S-attributed grammar and L-attributed grammar in semantic analysis?
5. Illustrate the storage organization memory in the perspective of compiler writer with neat diagram.
6. Explain the dynamic storage allocation techniques.
7. Develop a quicksort algorithm to reads nine integers into an array a and sorts them by using the concepts of activation tree.
8. Generate an intermediate code for the following code segment with the required syntax-directed translation scheme.

if (a > b)

x = a + b

else

x = a - b

1. Describe.in detail about

(i) Quadruples

(ii) Triples.

1. Evalute and draw a annotated parsing tree for given SDT arithmatic expression 4-6/3+5. Given grammar

E > E_1+T E > E_1-T E > T, T > T_1 * F T > T_1/F T > F F > E F > num.

1. Explain in detail about optimization of basic blocks and Construct the DAG for the following Basic block & explain it.
2. t1 : = 4 * i

3. t2:= a [t1]
4. t3: = 4 * i
5. t4:= b [t3]
6. t5:=t2*t4
7. t6:=Prod+t5
8. Prod:= t6
9. t7:= i+1
10. I:= t7
11. if i<= 20 goto (1).

Explain Global data flow analysis and Data structure for symbol table.

UNIT 4

1. How the activation record is pushed onto the stack. (Short Answer)
2. What is heap allocation? (Short Answer)
3. Discuss the main idea of Activation tree. (Short Answer)
4. Prepare optimal code sequence for the given sequence. (Short Answer)

t=a+b

t=t*c

t=t/d

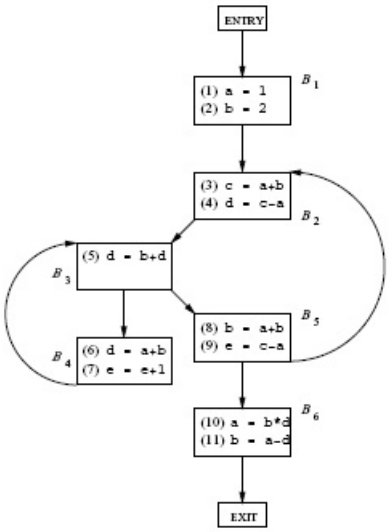
1. Illustrate the storage organization memory in the perspective of compiler writer with neat diagram.
2. Develop a quicksort algorithm to reads nine integers into an array a and sorts them by using the concepts of activation tree.
3. Define fragmentation? Describe in detail about how to reduce the fragment.
4. Define the heap management of memory and describe in detail about it .
5. Write a short note on:
6. copy propagation
7. Dead code Elimination
8. code motion
9. Managing and coalescing free space
10. Best fit and next object placement.

UNIT 5

1. How is liveness of a variable calculated? Identify it. (Short Answer)
2. Identify the constructs for optimization in basic block. (Short Answer)
3. Give the main idea of dead code elimination and constant folding. (Short Answer)
4. Point out the characteristics of peephole optimization. (Short Answer)
5. What is DAG? Point out advantages of DAG. Draw the DAG for the statement a= (a*b+c) – (a*b+c) and evaluate it.
6. Explain in detail about the data-flow schemas on basic block and the transfer equations for reaching definitions with example
7. (i).Explain in detail about optimization of basic blocks.

(ii).Construct the DAG for the following Basic block & explain it.

1. t1: = 4 * i
2. t2:= a [t1]
3. t3: = 4 * i
4. t4:= b [t3]
5. t5:=t2*t4
6. t6:=Prod+t5
7. Prod:=t6
8. t7:=i+1
9. i:= t7
10. if i<= 20 goto (1).
11. Write about the following in detail
12. copy propagation
13. Dead code Elimination
14. code motion
15. Illustrate the Iterative algorithm for reaching definitions
16. Discuss the live variable analysis
17. Illustrate the Iterative algorithm for reaching definitions
18. Discuss the live variable analysis
19. Use of Algebraic Identities.
20. Compute the grn and Kill sets for each Block
21. In and Out sets for each block
22. Compute e_gen and e_kill

23. 

24. Analyze the available expressions on the following code by
25. converting into basic blocks and compute global common sub expression elimination.
26. i = 0
27. a:= n-3
28. if i < a then loop else end
29. label loop
30. b:= i -4
31. c:= p + b
32. d:= M[c]
33. e:=d-2
34. f:=i-4
35. g:=p+f
36. m[g]:=e
37. i:=i+1
38. a:=n-3
39. if i < a then loop else end
40. label end
41. Identify the loops of the flow graph
42. Identify the global common sub expression for each loop
43. Identify Induction variables for each loop
44. Identify loop invariant computation for each loop
45.