Compiler Design
Unitwise Important Questions

Unit-1: Introduction, Lexical Analysis

Short answer Questions

1. Define preprocessor, compiler, assembler, linker, and loader (Language processors)
2. Differentiate compiler and interpreter
3. Define regular expression
4. Define token, pattern, lexeme with examples
5. What is a symbol table
6. Describe the languages generated by the following regular expressions
   i) 0*10*10*1 ii) (a+b)*abb
7. What is finite automata, Explain structure of compiler?
8. Differentiate DFA and NFA
9. Differentiate pass and phase of a compiler
10. What are analysis and synthesis phases of a compiler

Essay type Questions

** 1. Discuss the phases (structure) of a compiler indicating the inputs and outputs of each phase in translating the statement position:=initial+rate*60
2. Explain about the science of building a compiler
3. Explain about programming language basics for compiler design
* 4. Explain about the role of lexical analyzer in a compiler
5. Explain the input buffering scheme for scanning the source program. How the use of sentinels can improve its performance?
6. Explain about recognition of tokens
*7. What is LEX? Explain different sections of a LEX program with examples
8. Explain the procedure for constructing an NFA for the regular expression r=(a|b)*abb
9. Write an algorithm to convert a given NFA into equivalent DFA and give an example
10. Write down the steps in constructing DFA for the regular expression (a|b)*aab(a|b)*

Unit-2: Syntax Analysis

Short answer Questions

1. Define context-free grammar
2. What is parsing?
3. What is left most and right most derivation. Give examples
4. What is a parse tree? Give an example
5. What is an ambiguous grammar
6. Compare SLR, CLR, LALR
7. Differentiate top-down and bottom-up parsing
8. Define LL(1) grammar
9. Define the rules for FIRST and FOLLOW
10. What are the difficulties in top-down parsing?

Essay type Questions

1. What is an ambiguous grammar. Explain the procedure for eliminating ambiguity from a grammar with examples
   Step 1: remove left recursion from CFG
   Step 2: left factor the CFG

*2. What is left recursion? Describe the algorithm used for eliminating left recursion.
Eliminate left recursion from the following grammar
E→E+T | T, T→T*F | F, F→(E) | id

*3. What is left factoring? Describe the algorithm for left factoring a grammar with an example

*4. Construct predictive parsing table and verify whether the following grammar is LL(1) or not
E→E+T | T, T→T*F | F, F→(E) | id

*5. Write the algorithm for recursive descent parsing. Explain with an example

**6. Write the rules and compute FIRST and FOLLOW for the following grammar
E→E+T | T, T→T*F | F, F→(E) | id

((If difficult skip this

7. Construct SLR paring table for the following grammar
   E→E+T | T, T→T*F | F, F→(E) | id
8. Construct CLR paring table for the following grammar
   E→E+T | T, T→T*F | F, F→(E) | id
9. Construct LALR paring table for the following grammar
   E→E+T | T, T→T*F | F, F→(E) | id

))

10. What is YACC? Explain the different sections of a YACC program with examples
11. Explain about different LR parsers (SLR, CLR, LALR parsers)

Unit-3: Syntax-Directed Translation, Intermediate-Code Generation

Short answer Questions

1.How synthesized attributes differ from inherited attributes. Give examples.
2. Compare SDD and SDT
3. Differentiate S-attributed definition and L-attributed definition
4. List out the applications of Syntax-directed translation.
5. How to find evaluation order of SDDs?
6. What is coercion?
7. What is DAG? What are the applications of DAG?
8. What is a type expression? Give examples
9. Define type checking.
10.What is intermediate code
11.What is three address code

Essay type Questions

**1. Write SDD for a simple basic(desk) calculator and construct an annotated parse tree for the input expression (4*7+1)*2
2. Explain how SDTs are used for constructing syntax trees
**3. Write Quadruples, Triples, Indirect triples for the statement a:=b*-c+b*-c
(What are the three representations for implementing three address code. Explain with examples)
(Write triple representations for x:=y[i], x[i]:=y)
4. Write SDD for translating control flow statements into intermediate code
5. Write SDD for translating switch statement into intermediate code
6. What is a type checker? Give the specification of a simple type checker.
7. Construct DAG for the expression a+a*(b-c)+(b-c)*d

Unit-4: Run-Time Environments, CodeGeneration

Short answer Questions

1. What are the limitations of static allocation?
2. What is garbage memory?
3. Explain about run-time memory organization
4. Define basic block
5. Define flow graph
6. What is target language?
7. Define register allocation and assignment
8. What are the different object code forms (Assembly language code, Relocatable machine code, Absolute machine code)

Essay type Questions

*1. What is an activation record? Explain the fields of an activation record with a neat diagram
*2. Explain various storage allocation strategies and also mention their merits and demerits
(static allocation, stack allocation, heap allocation)
*3. Explain about stack storage allocation with an example
4. Explain about heap memory management
5. Explain about how to access nonlocal data on a stack by using access links and displays
*6. Briefly explain
a) Garbage collection (reference counting algorithm)
b) Trace based collection (Mark-and-sweep algorithm)
*7. Explain the issues in the design of a code generator
*8. Explain the different peephole optimization techniques with examples
**9. Write code generation algorithm and explain with an example
**10. Explain how to construct a flow graph for a given program with an example
**11. Write an algorithm for partitioning three address code into basic blocks and give an example
**12. Explain about the methods(strategies) for register allocation and assignment

(reserve registers for some specific values, allocate registers for global variables, usage count of a variable, register assignment for outer loops)

13. Explain about instruction scheduling with an example

*14. Explain how DAGs are used for optimizing basic blocks(Write a detailed description on DAG)

Unit-5: Machine-Independentcode Optimizations

Short answer Questions

1. Differentiate machine-dependent(language-independent) and machine-independent(language-dependent)code optimizations
2. Write the data-flow equations
3. What is common sub-expression elimination? Explain
4. What are induction variables? What is induction variable elimination?
5. Define live variable
6. What is dead code (dead variable)
7. What is local optimization and global optimization?

Essay type Questions

**1. Explain the principal sources of optimization with examples
(OR)Explain the function-preserving transformations (local optimization techniques) and loop optimization techniques with examples

2. Explain about partial-redundancy elimination with examples
3. * Explain about constant propagation with examples
4. **Explain about the basics of data-flow analysis(Explain various notations and equations used in data flow analysis)
5. Explain about the foundations of data-flow analysis
6. Explain about loops in flow graphs