COMPILER DESIGN

QUESTION BANK

(Dr. Kalyan Kumar Jena)

1.Explain why it is possible to design an independent Lexical Analyzer?
2.What are the drawbacks of SLR(1) Parser?
3.What are the various data structures used to create a symbol table?
4.What do you mean by porting of a compiler?
5.Why LR parsing is prefer over other parsers?(Advantages of LR Parser)
6.What is dead code elimination?
7.What is Reduce-Reduce and Shift-Reduce conflict in LR parser?
8.What are the goals of error handler in a parser?
9.What are the error recovery actions in a Lexical Analyzer?
10.What is the basic block in code segment? What is Back patching?
11.What is S-attributed grammar and L-attributed grammar in semantic analysis?
12.What do you mean by postfix translations?
13.Differentiate between NFA and DFA.
14.Define CFG. Explain ambiguity in CFG.
15.Explain about compile time and run time errors.
16.Describe the structure of LL parser and LR parser.
17.Distinguish between syntax and semantics of a programming language.
18.What are the main function of the different stages of a compiler?
19.What is Syntax Directed Translation Scheme(SDTS)?
20.What are the different forms of intermediate code used in compilation process?
21.Describe the use of Flow Graph in Compiler writing.
22.Why do most compilers contain symbol table?
23. Define Cross Compiler. Explain Bootstrapping.
24.Why common approach is being used to design the lexical analyzer for compilers?
25.Explain the Front end and Back end of a compiler.
26.What is LL(1) and SLR(1)Grammar?
27.What is LEX,YACC,FLEX and BISON?
28.What is a semantic error? Explain with an example.
29.Name the methods of code optimization.
30.What are the various problems in code generations?
31.What are the various data structures used in LR parser design?
32.Explain Top-down and Bottom-up parsing.
33.Define Token & Lexeme.Construct NFA for the RE:$(a|b)^*$abb,and convert it to DFA
34.Draw the state transition diagram for BEGIN,IF,THEN,ELSE,END

35Explain SDTS ,SDD , L-attributed& S-attributed grammar.
36 Explain in detail about symbol table.

37.Explain in detail about code optimization.
38.Explain the symbol table of a compiler for a block structured Language.
39.Explain in details about intermediate code generation.
40.Write Quadruples ,Triples & Indirect Triples for the following expression:(i)X[I]:=Y,X:=Y[I] (ii) -(a+b)*(c+d)-(a+b+c)
41.Describe Run Time Storage Allocation.
42.Discuss the construction of ACTION and GOTO table for SLR,CLR and LALR Parser.
43.What is an Activation record? Explain its different components.
44.Explain the various issues in the design of code generator.
45.What is DAG? Explain its algorithm. Construct the DAG for the following statements: T_1:=4*I,T_2:=addr(A)-4,T_3:=T_2[T_1],T_4:=4*I,T_5:=addr(B)-4, T_6:=T_5[T_4],T_7:=T_3*T_6,T_8:=PROD+T_7,PROD=T_8,T_9:=I+1,I:=T_9, if I<=20 goto (1)
46.Explain FIRST and FOLLOW algorithm.
47.Costruct the SLR Parse table for the following grammar: E->(L)|a, L->L,E|E Show the parsing action for the string ((a),a,(a,a))
48.Construct the Predictive parse table (LL parse table) for the following grammar: S->aBDh,B->cC, C->bc|€ , D->EF ,E->g|€ ,F->f|€
49.For the following grammar :E->E*E,E->E+E,E->(E),E->id show the various shift-reduce parsing action with respect to the input string :id+id*id
50.Construct the CLR table for the grammar: S->AaAb, A->BbBa, A->€, B->€
51.Construct the LALR table for the grammar:S->Aa|bAc|dc|bda , A->d
52.What are the necessity of optimization in compilation?Discuss the factors influencing optimization.
53.Define operator precedence relation and operator precedence grammar. Write down the steps for operator-precedence parser. Construct the mapped table for string " *id=id" and show its parsing action.
54.Explain the elimination of Left factoring and Left recursion with example.
55.Explain the goals of error handler. Explain Lexical phase errors and show how to eliminate such errors.
565.Generate Three address code for the following code segment: main() { int a=1; int b[10]; while(a<=10) b[a]=2*a; } and also specify tokens and lexemes for this segment.
57.What is annotated parse tree? Construct such tree for the following: (4+3)*(5+7)
58.Find out FIRST and FOLLOW for the following grammar: S->aAB|Ba|€ , A->aAb|€ , B->Bb|c

58. What is Compiler? Design the Analysis and Synthesis Model of Compiler.
59. Write down the five properties of compiler.
60. What is translator? Write down the steps to execute a program.
61. Discuss all the phases of compiler with a with a diagram.
62. Write a short note on:
YACC
Pass
Bootstrapping d.
LEX Compiler
Tokens, Patterns and Lexemes
63. Write the steps to convert Non-Deterministic Finite Automata (NDFA) into Deterministic Finite Automata (DFA).
64. What is Regular Expression? Write the regular expression for:

a. **R=R1+R2 (Union operation)**

b. **R=R1.R2 (concatenation Operation)**

c. **R=R1* (Kleen Clouser)**

d. **R=R+ (Positive Clouser)**

e. **Write a regular expression for a language containing strings which end with "abb" over Σ= {a,b}.**

f. **Construct a regular expression for the language containing all strings having any number of a's and b's except the null string.**

65. **Construct Deterministic Finite Automata to accept the regular expression : (0+1)* (00+11) (0+1)***

66. Derivation and Parse Tree:

Let G be a Context Free Grammar for which the production Rules are given below: S -> aB|bA

A -> a|aS|bAA B -> b|bS|aBB

Drive the string *aaabbabbba* using the above grammar (using Left Most Derivation and Right most

Derivation).

67. Explain the parsing techniques with a hierarchical diagram.
68. What are the problems associated with Top Down Parsing?
69. Write the production rules to eliminate the left recursion and left factoring problems.
70. Consider the following Grammar:
A-> ABd|Aa|a

B-> Be|b

Remove left recursion.

71. Do left factoring in the following grammar:
A-> aAB|aA|a
B-> bB|b
72. Write a short note on:

a. Ambiguity (with example)

b. Recursive Descent Parser

c. Predictive LL(1) parser (working)

d. Handle pruning

e. Operator Precedence Parser

73. . Write Rules to construct FIRST Function and FOLLOW Function.
74. Consider Grammar:
E-> E+T|T
T-> T*F|F
F-> (E)|id

Write the algorithm to create Predictive parsing table with the scanning of input string.

75. **Show the following**
    Grammar: S-> AaAb|BbBa

    A-> €

    B-> €

    Is LL(1) and parse the input string "ba"

76. **Consider the**
    grammar: E-> E+E

    E-> E*E

    E->id

Perform shift reduce parsing of the input string "id1+id2+id3".

77. **Write the properties of LR parser with its structure. Also explain the techniques of LR parser.**
78. **Write a short note on:**

a. **Augmented grammar**
b. **Kernel items**

c. **Rules of closure operation and goto operation**
d. **Rules to construct the LR(0) items**

79. **Consider the following**
    grammar:
    S-> Aa|bAc|Bc|bBa

    A-> d

    B-> d

Compute closure and goto.

80. **Write the rules to construct the SLR parsing table.**
81. **Consider the following**
    grammar: E-> E+T|T

    T-> TF|F

    F-> F*|a|b

Construct the SLR parsing table and also parse the input "a*b+a"

82. **Write the rules to construct the LR(1) items.**

83. What is LALR parser? Construct the set of LR(1) items for this grammar: S-> CC
C-> aC C->d

84. Show the following grammar

S->Aa|bAc|Bc|bBa

A->d

B->d

Is LR(1) but not LALR(1).

85. Write the comparison among SLR Parser, LALR parser and Canonical LR Parser.

86. What is syntax directed translation (SDD)?

87. Write short note on:

a. Synthesized attributes b.
Inherited attributes

c. Dependency graph

d. Evaluation order

e. Directed Acyclic Graph (DAG)

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

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

89. Differentiate between synthesized translation and inherited translation.

90. What is intermediate code and write the two benefits of intermediate code generation.

91. Write the short note on:

a. Abstract syntax tree

b. Polish notation

c. Three address code

d. Backpatching

92. Construct syntax tree and postfix notation for the following expression:

(a+(b*c)^d-e/(f+g)

93. Write quadruples, triples and indirect triples for the expression:
    -(a*b)+(c+d)-(a+b+c+d)
94. Write the three address statement with example for:

a. Assignment

b. Unconditional jump (goto)

c. Array statement (2D and 3D)

d. Boolean expression

e. If-then-else statement

f. While, do-while statement

g. Switch case statement

95. Write the definition of symbol table and procedure to store the names in symbol table.
96. What are the data structures used in symbol table?
97. What are the limitations of stack allocation?
98. Write two important points about heap management.
99. Write the comparison among Static allocation, Stack allocation and Heap Allocation with their merits and limitations.
100. What is activation record? Write the various fields of Activation Record.
101. What are the functions of error handler?
102. Write a short note on Error Detection and Recovery.
103. Classify the errors and discuss the errors in each phase of Compiler.
104. What are the properties of code generation phase? Also explain the Design Issues of this phase.
105. What are basic blocks? Write the algorithm for partitioning into Blocks.
106. Write a short note on:

a. Flow graph (with example)

b. Dominators

c. Natural loops

d. Inner loops

e. Reducible flow graphs

107. Consider the following program

```
code: Prod=0;

I=1; Do{

Prod=prod+a[i]*b[i]; I=i+1;

}while (i<=10);
```

a. Partition in into blocks
b. Construct the flow graph

108. What is code optimization? Explain machine dependent and independent code optimization.

109. What is common sub-expression and how to eliminate it? Explain with example.

110. Write a short note with example to optimize the code:

a. Dead code elimination

b. Variable elimination

c. Code motion

d. Reduction in strength

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

111. Define a compiler?

112. Describe the Analysis Synthesis Model of compilation.

113. Define a symbol table.

114. What are the functions of preprocessors?

115. Define Assembly Code.

116. Define tokens, Patterns and lexemes.

117. What are the possible error recovery actions in lexical Analyzer?

118. What are the three general approaches to the implementation of a Lexical Analyzer?

119. Define a Sentinel.

120. Describe briefly rational preprocessors with an example.

121. What are the reasons for separating the analysis phase of compiling into Lexical analysis and parsing?

122. What is the function of a loader?

123. Give the diagrammatic representation of a language processing system.

124. Explain briefly the producer consumer pair of a lexical analyzer and parser.

125. Mention the issues in a lexical analyzer.
126. Mention some of the cousins of the compiler.
127. What are rational preprocessors?
128. Explain in detail about the role of Lexical analyzer with the possible

error recovery actions.

129. (a)Describe the following software

tools i. Structure Editors

ii. Pretty printers

iii. Interpreters

(b)Write in detail about the cousins of the compiler.
130. Describe in detail about input buffering. What are the tools used for

constructing a compiler?

131. (a)Explain the functions of the Lexical Analyzer with its

implementation.(10) (b)Elaborate specification of tokens.

132. (a)What is a compiler? Explain the various phases of compiler in detail, with a neat sketch.
133. (b) Elaborate on grouping of phases in a compiler.(6)
134. (a) Explain the various phases of a compiler in detail. Also write down the output for the following expression after each phase a: =b*cd. (8)
135. (b) What are the phases of the compiler? Explain the phases in detail. Write down the output of each phase for the expression a: = b +c * 50. (8)
136. Draw a NFA for a*|b*.
137. What do you mean by Handle Pruning?
138. Define LR (0) items.
139. What do you mean by viable prefixes?
140. Define handle.
141. What are the algebraic properties of regular expressions?
142. What is finite automata?
143. What are the goals of error handler in a parser?
144. What is an ambiguous grammar? Give an example.
145. .What is phrase level error recovery?
146. What are the disadvantages of operator precedence parsing?
147. What is a predictive parser?
148. Eliminate left recursion from the following grammar A->Ac/Aad/bd/c.
149. What is LL (1) grammar? Give the properties of LL (1) grammar.
150. Give the algorithm for Left Factoring a Grammar.
151. What is Left Recursion? Give an example for eliminating the same.
152. What is FIRST and FOLLOW? Explain in detail with an example. Write

down the necessary algorithm.

153. Construct Predictive Parsing table for the following grammar:

S > (L)/ a

S->(L)/a

L> L, S/S

and check whether the following sentences belong to that grammar or not.

(i) (a,a)

(ii) (a, (a , a))

(iii) (a,((a,a),(a,a)))

154. (a) Construct the predictive parser for the following

grammar: S > (L)|a L > L,S|S.

(b) Construct the behaviour of the parser on sentence (a, a) using the grammar: S >

(L)|a L > L,S|S.

155. 4. (a) 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

(b) For the grammar given below, calculate the operator precedence relation and the precedence functions.
E>E+E|E –E|E*E|E/E|E^E|(E)| E|id

156. Check whether the following grammar is a LL(1) grammar

S > iEtS | iEtSeS | a E> b

Also define the FIRST and FOLLOW procedures.

157. (a) Consider the grammar given

below. E>E+T E>T T>T*F T>F

F>(E)

F>id. Construct an LR Parsing table for the above grammar. Give the moves of LR

parser on id*id+id.

(b) What is a shiftreduce parser? Explain in detail the conflicts that may occur during shiftreduce parsing. (4)

158. How would you represent the following equation using the

DAG, a: =b*c + b*c. What is the purpose of DAG?

159. What is the intermediate code representation for the expression a or b and not c ?

160. How would you map names to Values?
161. What are the various methods of implementing three address statements?
162. Suggest a Suitable approach for completing hash function.
163. What are the methods of representing a syntax tree?
164. Give the Syntax directed definition of if else statement.
165. What is back patching?
166. What are the applications of DAG?
167. Define marker non terminals with an example.
168. Why are quadruples preferred over triples in an optimizing Complier?
169. Give the triple representation of a ternary operation x:= y[i]
170. Give the Semantic rules for the production S _ while E do S1.
171. Let A be a 10x20 array with low_1=low_2=1. Therefore n_1=10 and n_2=20.

Take w to be 4. Give the annotated parse tree for the assignment x:=A[y,z]

172. What is short circuit or jumping code?
173. How would you generate the intermediate code for the flow of

control statements?Explain with examples.

174. (a)What are the various ways of calling procedures? Explain in detail.

(b)What is a three address code? Mention its types. How would you implement

the three address statements? Explain with examples.

175. How would you generate intermediate code for the flow of control statements? Explain with examples.
176. (a)Describe the method of generating syntax directed definition for

Control statements.

(b)Give the semantic rules for declarations in a procedure.

177. 5. (a) How Back patching can be used the generate code for Boolean expressions and flow of control statements.

(b)Explain how the types and relative addresses of declared names are computed and how scope information is dealt with.

178. 6. (a) Describe in detail the syntaxdirected translation of case

statements.

(b) Explain in detail the translation of assignment statements.

179. How would you calculate the cost of an instruction?
180. What is an activation record for a procedure?
181. What is a Basic Block?
182. What are the steps involved in partitioning a Sequence of three address statements into basic blocks?
183. What are the limitations of static allocation?

184. What is dead code elimination?
185. Give the primary structure preserving transformations on Basic Blocks.
186. Draw the diagram of the general activation record and give the purpose of any two fields.
187. What is stack allocation?
188. What are dags and how are they useful in implementing transformations on basic blocks?
189. What is peephole optimization?
190. Mention the transformations that are characteristic of peephole optimizations.
191. What are machine idioms?
192. Give the applications of dags.
193. 1What are register descriptors?
194. Briefly describe address descriptors.
195. (a)Explain the issues in design of code generator.

(b)Explain peephole optimization.

196. (a)Discuss run time storage management of a code generator.

(b)Explain DAG representation of the basic blocks with an example.
197. (a)Explain the simple code generator with a suitable example.

(b)Describe about the stack allocation in memory management.

198. (a) What are the different storage allocation strategies?

(b)What are steps needed to compute the next use information?

199. (a) Write detailed notes on Basic blocks and flow graphs.

(b)How would you construct a DAG for a Basic block? Explain with an example.

200. What are called optimizations and what is an optimization compiler?
201. Mention the criteria for code improving transformations.
202. Mention the function preserving, code improving transformations.
203. What is code motion? Give an example.

204.What is constant folding?
205.What are induction variables? What is induction variable elimination?
206. What is a cross compiler? Give an example.
207.What are the properties of optimizing compilers?
208.Mention the different storage allocation strategies.
209.Mention the limitations of static allocation.
210.What are calling sequences and give brief notes on its types.
211.When does a dangling reference occur? Give its impact on programs.
212.Give the situations in which stack allocation can not be used.
213.Mention the different types of parameter passing.
214.What are the two approaches of implementing Dynamic Scope? Give the difference between the two.
215.(a) Explain the principle sources of optimization in

detail.

216. (b)What are the various ways of calling procedures?

217.2 (a) Discuss about the following:

i). Copy Propagation

ii) Deadcode Elimination and

iii)Code motion

(b)Describe in detail about

the stack allocation in

memory management.

(a) Write about Data flow

analysis of structural

programs.

(b)Describe the various

storage allocation strategies

218. (a) Describe in detail the source language

issues.

(b) Explain in detail access to non local names.

219.(a) Elaborate storage organization.

(b)Write detailed notes on parameter passing.

220.(a) Explain optimization of basic blocks.

(b) Explain the various approaches to compiler development.