

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University) Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

MUST KNOW CONCEPTS



MKC



Subject		16ITD08/Principles of Compiler Design			
S. No. Term		Notation (Symbol)	Concept/Definition/Meaning/Units/Equation/ Expression	Units	
	UNIT-I INTRODUCTION TO AUTOMATA AND COMPILER				
1	Translator		It converts source language into target language		
2	Compiler		System software which translates source program into target program		
3	Interpreter		System software which accepts source program line by line and produces target program		
4	Assembler		converts assembly language into machine code		
5	Loader		A loader is a program that places machine code of the programs into memory for execution		
6	Link-editor		The linker links the code in one file which may refer to a location in another file		
7	Two Parts of Compilation		Analysis and Synthesis		
8	Analysis (Front end of Compiler)	Analysis part breaks the source program into constituent pieces and creates an intermediate representation			
9	Synthesis (Back end of Compiler)		Synthesis part takes the intermediate representation as input and transforms it to the target program.		
10	6 Phases of Compiler		1.Lexical analysis 2. Syntax analysis 3. Semantic analysis 4. Intermediate code generation 5. Code optimization 6. Code generation		
11	Lexical analysis (Scanner)		It accepts lexemes which produces token as output		
12	Token		Sequence of characters that can be treated as a single logical entity. Eg: Number, Identifiers ,keywords , etc		
13	Lexeme	Sequence of characters in the source program			
14	Pattern	Set of strings is described by a rule called a pattern associated with the token.			
15	Symbol Table		Data structure that contains a record for each symbol		
16	Syntax analysis (Parser)	accepts sequence tokens as input and produces parse tree as output			



17Intermediate representation 2 Easy to translate into the target machine18Properties of code $>$ Three address code have atmost 3 operand19Use of Code Optimization $>$ Atmost 1 operator additional to = $>$ 20Goals of Error HandlerReport the presence of errors clearly and accurately.20Goals of Error HandlerReport the presence of errors clearly and accurately.21four common error-recovery $>$ 22Panic mode $>$ 23four common error-recovery $>$ 24four common error-recovery $>$ 25Panic mode $>$ 26Deterministic four Automate $>$ 27Panic mode $>$ 28Statement level. synchronizing token is found $>$ 29Panic mode $>$ 20Deterministic for a particular input character, the machine goes to or state only and null (or ε) move is not allowed $>$ 29Nondeterministic Finitei. a transition ii.move any number of states for a input.20LEXLexical Analysis21Erx (Context-free Grammata(NFA)i. a transition ii.move any number of states for a input.21Paricei. a transition ii.move any number of states for a input.22Panic and for a particular input character, the machine goes to or state only and null (or ε) move is not allowed23Secondary task of LAIncerta station ii.move any number of states for a input. <th></th> <th>Two properties</th> <th>It should be easy to produce</th> <th></th>		Two properties	It should be easy to produce	
17 representation \sim \sim 18 Properties of Three address \sim Three address code have atmost 3 operand 19 Use of Code Optimization Produces Faster Shorter code, target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. 20 Goals of Error Handler Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common error-recovery \rightarrow Statement level. Statement level. 22 Panic mode Sincard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis \rightarrow Simplicity of design 24 Error inistic Finite Automata (DFA) It consists of 5 tuples {Q, \sum , q, F, δ }. 25 Finite Automata (DFA) i. ε transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 27 Primary task of LA Eliminating white spaces.Comments LA 28 Primary task of LA Eliminating white spaces.Comments 29 Scondary task of LA Eliminating white spaces.Comments 29 Scontrext-free Grammars G= (V.P.S.	Intermediate Easy to translate into the target mach			
18 Properties of Three address code > Three address code have atmost 3 operand 19 Use of Code Optimization Produces Faster ,Shorter code,target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. 20 Goals of Error Handler Report the presence of errors clearly and accurately. 21 four common four common parser > Panic mode. 22 Panic mode > Statement level. 23 Issues in Lexical analysis > Compiler productions. 24 Panic mode Simpler efficiency is improved books of the compiler protability is enhanced 23 Issues in Lexical analysis > Compiler efficiency is improved books of 5 tuples {Q, Z, q, F, 8}. 24 Deterministic Finite Automata (DFA) It consists of 5 tuples {Q, Z, q, F, 8}. 25 Nondeterministic Automata(NFA) i. c transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 1. Alias name for LA 28 Primary task of LA Token generation 1. Alias name for LA 2. Context-free Grammars 29 Scondary task of LA Eliminating white spaces,Comments 1. Alias name for LA Eliminating white spaces,Comments<	17		Easy to translate into the target machine	
18 Properties of Three address code → Atmost 1 operator additional to = → Temporary variable used to store intermediate result 19 Use of Code Optimization Produces Faster .Shorter code,target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common error-recovery strategies in → Panic mode. 22 Panic mode Sitatement level. 23 Issues in Lexical analysis → Sitatement level. 24 Finite Automata (DFA) → Simplicity of desiga → Compiler efficiency is improved → Compiler efficiency is inproved → Compiler efficiency is not allowed 24 Deterministic Finite Automata (DFA) It consists of 5 tuples {0, ∑, q, F, δ}. 25 Finite Automata (DFA) i. ε transition ii.move any number of states for a input. 26 LEX Lexical Analysis or scanner 27 Alias name for LA Linear analysis or scanner 28 Primary task of LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Cortext-free Grammars Ge {(V,P.S,T) Tokes of replacing left most non-terminal by		representation	Three address and a have atmost 2	
18 Three address code > Atmost 1 operator additional to = 19 Use of Code Optimization Produces Faster ,Shorter code,target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. 20 Goals of Error Handler Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common four common errection programs > Panic mode. 22 Panic mode Sitatement level. 23 Issues in Lexical analysis > Sinplicity of design 24 Frinte Automata for a particular input character, the machine goes to one state only and null (or ε) move is not allowed 24 Finite Automata for a particular input character, the machine goes to one state only and null (or ε) move is not allowed 25 Finite Automata (NFA) Linear analysis or scanner 26 LEX Lexical Analyzer Tool 27 Atias name for Lexical analysis or scanner Eiminating white spaces,Comments 30 Context-free G = (V.P.S.T) Granmars G = (V.P.S.T) 31 Derivation Fright side of production Fright side of production 32 Izet of crivation Fri		Duamantias of		
code > Temporary variable used to store intermediate result 19 Use of Code Optimization Produces Faster, Shorter code,target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 20 Four common > Panic mode. 21 four common > Panic mode. 22 Panic mode discard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design 24 Finite Automata (OFA) It consists of 5 tuples (Q, Q, q, F, \delta). Finite Automata (OFA) for a particular input character, the machine goes to one state only and null (or e) move is not allowed 25 Nondeterministic Finite Automata (NFA) i. transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analyzer Gol I. 28 Pinary task of Fining task of Concess of replacing the non-terminal by its right side of production 29 LEX Lexical Analyzer Tool 29 Context-free Grammars G= (V.P.S.T) 30	10	1	-	
Image: constraint of the second state of the seco	18			
19 Use of Code Optimization Produces Faster .Shorter code, target code that consumes less power 20 Goals of Error Handler Report the presence of errors clearly and accurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common error-recovery strategies in parser > Panic mode. > Statement level. 22 Panic mode Synchronizing token is found 23 Issues in Lexical analysis > Global corrections. > Simplicity of design 24 Deterministic Finite Automata (DFA) It consists of 5 tuples {(O, $\sum, Q, F, \delta)$. for a particular input character, the machine goes to one state only and null (or c) move is not allowed 25 Finite Automata(NFA) i. ternsition ii.move any number of states for a input. 26 LEX Lexical Analyzer Token generation 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {(V,P,S,T] 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of Left derivation Process of replacing left most non-terminal by its right side of production 33 Le		code		
19 Optimization consumes less power 10 20 Goals of Error Handler Report the presence of errors clearly and accurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common error-recovery strategies in parser > Panic mode. Statement level. Statement level. Statement level. Strategies in parser 22 Panic mode discard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design > Compiler portability is enhanced 24 Deterministic Finite Automata (DFA) It consists of 5 tuples (Q, \sum, q, F, δ) . for a particular input character, the machine goes to one state only and null (or ε) move is not allowed 25 Nondeterministic Finite Automata (DFA) i. curasition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 1 27 Lexical Analyzis Linear analysis or scanner 1 28 Primary task of LA Token generation 1 29 Sccondary task of LA Eliminating white spaces,Comments 1 30 Context-free Grammas G= {V,P,S,T} 1				
OptimizationConsumes less power20Goals of Error HandlerReport the presence of errors clearly and accurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs21four common error-recovery strategies in parser> Panic mode. > Statement level. > Error productions. > Global correction22Panic mode> Simplicity of design > Simplicity of design > Compiler portability is enhanced23Issues in Lexical analysis> Simplicity of design > Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples {Q, Σ , q, F, δ }. for a particular input character, the machine goes to one state only and null (or e) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Alias name for LAEliminating white spaces,Comments28Prinay task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing left most non-terminal by its right side of production33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-termina	19			
20Goals of Error Handleraccurately. Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs21four common error-recovery strategies in parser> Panic mode. > Statement level. > Statement level. > Statement level.22Panic mode> Statement level. synchronizing token is found23Issues in Lexical analysis> Simplicity of design > Compiler efficiency is improved > Compiler efficiency is improved > Compiler optrability is enhanced24Finite Automata (DFA)It consists of 5 tuples (Q, \sum , q, F, 8). for a particular input character, the machine goes to one state only and null (or ε) move is not allowed i. i. a transition ii.move any number of states for a input.25Nondeterministic Finite Automata(NFA)Linear analysis or scanner26LEXLexical Analyzer Tool27Alias name for LALinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammasG= {V,P,S,T}31DerivationProcess of replacing left most non-terminal by its right side of production33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing a string by an Non terminal according to a grammar production		Optimization		
20 Goals of Error Handler Recover from each error quickly enough to detect subsequent errors. Add minimal overhead to the processing of correcting programs 21 four common error-recovery strategies in parser > Panic mode. Statement level. 22 Panic mode > Statement level. Statement level. 23 Issues in Lexical analysis > Simplicity of design Compiler efficiency is improved > Compiler efficiency is improved > Compiler efficiency is inproved > Compiler efficiency is indicated to the state only and null (or c) move is not allowed 24 Deterministic Finite Automata (DFA) It consists of 5 tuples {Q, \sum , q, F, \delta}. 25 Nondeterministic Finite Automata (DFA) i. ε transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 1 27 Aius name for LA Linear analysis or scanner 1 28 Primary task of LA Eliminating white spaces,Comments 1 30 Context-free Grammars G= {V,P,S,T} 3 1 31 Derivation Process of replacing the non-terminal by its right side of production 1 2 32 Types of derivation Left derivation Process of replacing right most non-ter				
20Handlerdetect subsequent errors. Add minimal overhead to the processing of correcting programs21four common error-recovery strategies in parser \geq Panic mode. \leq Statement level. \leq Eirror productions. \geq Giobal correction22Panic modediscard tokens one at a time until a synchronizing token is found23Issues in Lexical analysis \geq Simplicity of design \geq Compiler portability is enhanced24Eeterministic Finite Automata (DFA)It consists of 5 tuples $\{Q, \Sigma, q, F, \delta\}$. for a particular input character, the machine goes to one state only and null (or ε) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Ahias name for LALinear analysis or scanner28Primary task of LAEliminating white spaces,Comments30Context-free GrammarsG= (V,P,S,T)31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing a string by an Non terminal according to a grammar production			•	
Handlerdetect subsequent errors. Add minimal overhead to the processing of correcting programs21four common error-recovery strategies in parser> Panic mode. > Statement level. > Error productions. > Global correction22Panic modediscard tokens one at a time until a synchronizing token is found23Issues in Lexical analysis> Simplicity of design > Compiler efficiency is improved > Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples {Q, \sum , q, F, δ }. for a particular input character, the machine goes to one state only and null (or ε) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Alias name for Lexical AnalyzisLinear analysis or scanner28Primary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing left most non-terminal by its right side of production32Types of derivationLeft derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing a string by an Non terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal by its right side of production	20			
correcting programs21four common error-recovery strategies in parser22Fanic mode23Issues in Lexical analysis24Finite Automata (DFA)25Deterministic Finite Automata (DFA)26LEX27Nondeterministic Finite Automata (DFA)28Nondeterministic Finite Automata (DFA)29Nondeterministic Finite Automata (DFA)20LEX21Lexical Analysis22Deterministic Finite Automata (DFA)23Issues in Lexical analysis24Deterministic Finite Automata (DFA)25Nondeterministic Finite Automata(NFA)26LEX27LEX28Primary task of LA29Context-free Grammars30Context-free Grammars31Derivation32Types of derivation33Left derivation34Right derivation35Reduction36Reduction37Right derivation38Reduction39Context-free Grammars31Derivation33Left derivation34Right derivation35Reduction36Reduction37Process of replacing a string by an Non terminal by its right side of production34Right derivation35Reduction36Process of replacing a string by an Non te	20	Handler		
21four common error-recovery strategies in parser> Panic mode. Statement level. > Statement level. Stratement level.<			Add minimal overhead to the processing of	
21 error-recovery strategies in parser > Statement level. > Error productions. > Global correction 22 Panic mode discard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design > Compiler portability is enhanced 24 Deterministic Finite Automata (DFA) It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. for a particular input character, the machine goes to one state only and null (or e) move is not allowed 25 Nondeterministic Finite Automata(NFA) i. c transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of I.A Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 33 Left derivation Process of replacing right most non-terminal by its right side of production 33 Left derivation Process of replacing right most non-terminal by its right side of production 34 Right derivation Process of replacing right most non-terminal by its right side of production				
21 strategies in parser > Error productions. 22 Panic mode discard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design 24 Issues in Lexical finite Automata (DFA) > Compiler portability is enhanced 24 Deterministic Finite Automata (DFA) It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. 25 Finite Automata (NFA) i. ϵ transition 26 LEX Lexical Analysis UNIT-II LEXICAL ANALYSIS 26 LEX Lexical Analysis 27 Alias name for Lexical Analysis Linear analysis or scanner 27 Alias name for LA Token generation 28 Primary task of LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing left most non-terminal by its right side of production 33 Left derivation Process of replacing left most non-terminal by its right side of production 33 Left derivation Process of replacing a string by an Non terminal according to a grammar production <td></td> <td>four common</td> <td>Panic mode.</td> <td></td>		four common	Panic mode.	
strategies in parser > Error productions. 22 Panic mode discard tokens one at a time until a synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design 24 Deterministic Finite Automata (DFA) It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. 24 Finite Automata (DFA) for a particular input character, the machine goes to one state only and null (or e) move is not allowed 25 Nondeterministic Finite Automata(NFA) i. ϵ transition ii.move any number of states for a input. UNTI-II LEXICAL ANALYSIS 26 28 Primary task of Linear analysis or scanner 28 Primary task of Eliminating white spaces,Comments 10 29 Secondary task of Linear analysis of production 10 29 Orienter of production 10 24 Detroited to production 25 Nondeterministic Finite Automata(NFA) 26 LEX	21	error-recovery	Statement level.	
22Panic modediscard tokens one at a time until a synchronizing token is found23Issues in Lexical analysis> Simplicity of design > Compiler efficiency is improved > Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. for a particular input character, the machine goes to one state only and null (or ε) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAEliminating white spaces,Comments29Secondary task of GrammarsEliminating white spaces,Comments30DerivationProcess of replacing the non-terminal by its right side of production33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	21	strategies in	Error productions.	
22 Panic mode synchronizing token is found 23 Issues in Lexical analysis > Simplicity of design 24 Issues in Lexical individual indininditi indindindia individua individual individual indi		parser		
23Issues in Lexical analysis> Simplicity of design > Compiler portability is enhanced23Issues in Lexical analysis> Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. for a particular input character, the machine goes to one state only and null (or e) move is not allowed25Nondeterministic Finite 	22		discard tokens one at a time until a	
23Issues in Lexical analysis \rightarrow Simplicity of design \rightarrow Compiler efficiency is improved \rightarrow Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples {Q, \sum , q, F, δ }. for a particular input character, the machine goes to one state only and null (or ε) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	22	Panic mode	synchronizing token is found	
23 Issues in Lexical analysis > Compiler efficiency is improved 24 Deterministic Finite Automata (DFA) It consists of 5 tuples { Q, Σ, q, F, δ }. for a particular input character, the machine goes to one state only and null (or e) move is not allowed 25 Nondeterministic Finite Automata(NFA) i. e transition ii.move any number of states for a input. 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LAA Token generation 29 Secondary task of LAA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Left derivation 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing a string by an Non terminal according to a grammar production	 1		
analysis➤ Compiler portability is enhanced24Deterministic Finite Automata (DFA)It consists of 5 tuples {Q, ∑, q, F, δ}. for a particular input character, the machine goes to one state only and null (or ɛ) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation33Left derivationProcess of replacing the most non-terminal by its right side of production34Right derivationProcess of replacing a string by an Non terminal according to a grammar production	23		1 7 8	
24Deterministic Finite Automata (DFA)It consists of 5 tuples $\{Q, \sum, q, F, \delta\}$. for a particular input character, the machine goes to one state only and null (or ε) move is not allowed25Nondeterministic Finite Automata(NFA)i. ε transition ii.move any number of states for a input.26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing left most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production		analysis		
24Deterministic Finite Automata (DFA)for a particular input character, the machine goes to one state only and null (or ɛ) move is not allowed25Nondeterministic Finite Automata(NFA)i. ɛ transition ii.move any number of states for a input.UNIT-II LEXICAL ANALYSIS26LEXLexical Analyzer Tool27Alias name for LALinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V.P.S.T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation33Left derivationProcess of replacing right most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production				
21 International (DFA) to one state only and null (or ɛ) move is not allowed 25 Nondeterministic Finite Automata(NFA) i. ɛ transition ii.move any number of states for a input. UNIT-II LEXICAL ANALYSIS 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LA Token generation 29 Secondary task of Context-free Grammars Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Process of replacing left most non-terminal by its right side of production 33 Left derivation Process of replacing right most non-terminal by its right side of production 34 Right derivation Process of replacing a string by an Non terminal according to a grammar production		Deterministic		
(DT1) allowed 25 Nondeterministic Finite Automata(NFA) i. ɛ transition ii.move any number of states for a input. 26 LEX UNIT-II LEXICAL ANALYSIS 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Process of replacing left most non-terminal by its right side of production 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing right most non-terminal by its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production	24	Finite Automata		
25Nondeterministic Finite Automata(NFA)i. ɛ transition ii.move any number of states for a input.UNIT-II LEXICAL ANALYSIS26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production		(DFA)		
25Finite Automata(NFA)1. E transition ii.move any number of states for a input.26LEXUNIT-II LEXICAL ANALYSIS26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of 			allowed	
25 Finite Automata(NFA) ii.move any number of states for a input. 26 LEX UNIT-II LEXICAL ANALYSIS 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Left derivation and Right derivation 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing right most non-terminal by its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production		Nondeterministic	i stransition	
Automata(NFA) L UNIT-II LEXICAL ANALYSIS 26 LEX Lexical Analyzer Tool 27 Alias name for Lexical Analysis Linear analysis or scanner 28 Primary task of LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Left derivation and Right derivation 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing a string by an Non terminal according to a grammar production	25			
26LEXLexical Analyzer Tool27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing right most non-terminal by its right side of production34Right derivationProcess of replacing a string by an Non terminal according to a grammar production		Automata(NFA)	nimove any number of states for a input.	
27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production			UNIT-II LEXICAL ANALYSIS	
27Alias name for Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free 	26	LEX	Lexical Analyzer Tool	
27Lexical AnalysisLinear analysis or scanner28Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production		Alios nome for		
128Primary task of LAToken generation29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	27		Linear analysis or scanner	
28 LA Token generation 29 Secondary task of LA Eliminating white spaces,Comments 30 Context-free Grammars G= {V,P,S,T} 31 Derivation Process of replacing the non-terminal by its right side of production 32 Types of derivation Left derivation and Right derivation 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing a string by an Non terminal according to a grammar production				
29Secondary task of LAEliminating white spaces,Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	28	-	Token generation	
29LAEliminating white spaces, Comments30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production				
30Context-free GrammarsG= {V,P,S,T}31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	29	•	Eliminating white spaces, Comments	
30GrammarsG= { V,P,S,1 }31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production				
31DerivationProcess of replacing the non-terminal by its right side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	30		$G = \{V, P, S, T\}$	
31Derivationright side of production32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production			Process of replacing the non-terminal by its	
32Types of derivationLeft derivation and Right derivation33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	31	Derivation		
32 derivation Left derivation 33 Left derivation Process of replacing left most non-terminal by its right side of production 34 Right derivation Process of replacing right most non-terminal by its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production		Types of		
33Left derivationProcess of replacing left most non-terminal by its right side of production34Right derivationProcess of replacing right most non-terminal by its right side of production35ReductionProcess of replacing a string by an Non terminal according to a grammar production	32		Left derivation and Right derivation	
35 Left derivation right side of production 34 Right derivation Process of replacing right most non-terminal by its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production		uenvation		
34 Right derivation Process of replacing right most non-terminal by its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production	33	Left derivation		
34 Right derivation its right side of production 35 Reduction Process of replacing a string by an Non terminal according to a grammar production				
35 Reduction Its right side of production Process of replacing a string by an Non terminal according to a grammar production	34	Right derivation		
according to a grammar production	~ '			
according to a grammar production	35	Reduction		
36 Alias of reduction reverse of derivation			according to a grammar production	
	36	Alias of reduction	reverse of derivation	
	50			

37	Yield	Leaf nodes of parse tree are concatenated from left to right to form the input string derived from a grammar	
38	Alias of yield	frontier	
39	Role of LEX	LEX translates a set of regular expression specifications into a C implementation of a corresponding finite state machine	
40	No. of sections in Lex program	three	
41	3 Sections	Declarations, Rules and Auxiliary functions	
42	Parser	parser takes input in the form of sequence of tokens and produces output in the form of parse tree	
43	Alias name for Syntax analysis	Hierarchical analysis or parsing	
44	No. of types of Parser	2	
45	Types of Parser	Top-down parser Bottom-up parser	
46	2 subtypes of topdown parser	Recursive descent parser Predictive parser	
47	Top-down parser	Parser builds parse tree from Root to Leaves	
48	Bottom-up parser	Parser builds parse tree from Leaves to Root	
49	Alias for predictive parser	Table driven parser or LL(1) Parser	
50	Conditions for Top down	Eliminating Left recursion & ambiguity Left factoring out Requires Backtacking	
		UNIT-III SYNTAX ANALYSIS	
51	Parse tree	A parse tree is a graphical representation of a derivation	
52	Properties of parse tree	root is labeled with Start symbol leaf is labeled with a token interior node is labeled by a non-terminal	
53	Ambiguous	A Grammar that produces more than one parse tree for some sentence using left most derivation / right most derivation	
54	Universal parsers	Cocke-Younger-Kasami algorithm and Earley's algorithm can parse any grammar	
55	Eliminating Left- Recursion	A context free grammar is said to be left recursive if it has a non terminal A with two productions in the following form $A \longrightarrow \beta A'$ $A \longrightarrow A' \longrightarrow \varepsilon \alpha A'$	
56	Left factoring	process of factoring out the common prefixes of two or more production alternates for the same non-terminal	
57	Handle	A substring that matches the right side of a production called handle	

58	Handle pruning	Applying the production to the substring results	
59	Alias of Bottomup	in a right-sentential form. rightmost derivations in reverse	
	parser		
60	SR parser	Shift Reduce parser	
61	4 operations in SR	Shift,Reduce, Accept and Error	
62	Shift	moving of symbols from input buffer onto the stack	
63	Reduce	RHS of production rule is popped out of stack and LHS of production rule is pushed onto the stack	
64	accept	successful parsing is done	
65	error	parser can neither perform shift action nor reduce action and not even accept action.	
66	Operator grammar	No Epsilon production and consecutive Non terminals	
67	Operator precedence parser	Bottom-up parser that interprets on operator grammar	
68	Precedence relations	a > b \rightarrow "a" has the higher precedence than terminal "b". a < b \rightarrow "a" has the lower precedence than terminal "b". a \doteq b \rightarrow "a" and "b" both have same precedence.	
69	LR(K) parser	"L" stands for left-to-right scanning of the input. "R" stands for constructing a right most derivation in reverse. "K" is the number look ahead symbols	
70	SLR parser	Simple LR parser	
71	CLR parser	Canonical LR parser	
72	LALR parser	Lookahead LR parser	
73	Most powerful parser	Canonical LR parser	
74	Advantages of OPP	 simplicity. easy to construct. Powerful that can be used for the programming language expressions 	
75	Disadvantages of OPP	 grammar of small class difficult to identify or decide that grammar recognized which language. not capable of handling the unary minus. 	
	UNIT-J	IV INTERMEDIATE CODE GENERATION	
76	Advantages of Machine independent intermediate form	Retargeting is facilitated. A machine independent code optimizer can be applied.	
77	Types of Intermediate languages	 Syntax Tree. Postfix Notation. Three Address code. 	

78	Syntax tree	condensed form of parse tree.	
79	DAG	Directed Acyclic Graph	
80	Postfix notation	Traverse left child,right child,and root	
81	General form of Three address code	X:=Y op Z	
82	Implementation of Three address code	 Quadruple. Triples Indirect Triples 	
83	Quadruples	Quadruples has four fields: op,arg1, arg2 and result.	
84	Triples	Triples has Three fields: op, arg1 and arg2	
85	Indirect triples	In addition to triples use a list of pointers.	
86	Pros of quadruples	Easy to rearrange code for global optimization	
87	Cons of quadruples	Lots of temporaries	
88	Use of Boolean expression	Alter the flow of control. Compute logical values	
89	Back patching	activityof filling up unspecified information of labels using appropriate semantic actions in during the code generation process.	
90	Functions of back patching	makelist(i) Merge(p1,p2) Backpatch(p,i)	
91	M.quad	M.quad records the number of the first statement of E2.code.	
92	Declaration	The process of declaring keywords, procedures, functions, variables, and statements with proper syntax	
93	Intermediate code generation	interface between front end and back end in a compiler	
94	DAG definition	tool that depicts the structure of basic blocks, helps to see the flow of values flowing among the basic blocks	
95	Use of DAG	DAG provides a good way to determine the common sub-expression.	
96	Procedure	A procedure returns the control but not any value to calling function or code.	
97	Function	A function returns a value and control to calling function or code.	
98	Calling function	Calling function contains the input (the actual parameters) which is given to the called function	
99	Called function	called function which then works on them because it contains the definition, performs the procedure specified and returns if anything is to be returned.	
100	Assignment statement	Assignment statements enable the programmer to define or redefine a symbol by assigning it a	

		value		
	UNIT-V CO	DE OPTIMIZATION AND CODE GENERATION		
101	Directed Acyclic Graph	DAG is similar to syntax tree but identify the common sub expression		
102	Basic block	sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt		
103	Dead code	Dead (or useless) code statements that compute values that never get used.		
104	Flow graph	A graph representation of three-address statements in which Nodes are basic block, and the edges represent the flow of control		
105	Copy Propagation	process of replacing the occurrences of targets of direct assignments with their values y=x $z=3+y \rightarrow z=3+x$		
106	Constant Folding	Deducing at compile time that the value of an expression is a constant and using the constant instead		
107	Code Motion	Modification that decreases the amount of code in a loop		
108	Reduction In Strength	replacing an expensive operation by a cheaper one		
109	Absolute machine language	program can be placed in a location in memory and immediately executed		
110	Relocatable machine language	program allows subprograms to be compiled separately. A set of relocatable object modules can be linked together and loaded for execution by a linking loader		
111	Loop unrolling (loop unwinding)	Loop unrolling increases the program's speed by eliminating loop control instruction and loop test instructions.		
112	Classes of local transformations	 structure-preserving transformations algebraic transformations. 		
113	Structure- preserving transformations	 common sub-expression elimination dead-code elimination renaming of temporary variables interchange of two independent adjacent statements 		
114	Inner Loop	A loop that contains no other loops is called an inner loop		
115	Local common sub-expression.	An occurrence of an common sub-expression within a block		
116	Global common sub-expression	An occurrence of an common sub-expression between the blocks	between the blocks	
117	Addressing mode	Way in which location of the operand may be specified		
118	Immediate addressing	Operand value should be specified as part of instruction. (#)		
119	Indirect addressing	address specified in the instruction are themselves an address (@)		
120	Register descriptor	register descriptor containing the list of variables currently stored in this register		

121	Address descriptor	address descriptor containing the list of locations	
122	Getreg function	where this variable is currently stored determine the location L where the result of the	
computation y op z should be stored		computation y op z should be stored	
123	Types of Jump	Conditional jump UnConditional jump	
123	statement		
		Optimization is a program transformation	
124	Code optimization	technique, which tries to improve the code by	
124	Code optimization	making it consume less resources and deliver	
		high speed.	
125	Code generation	Last phase is used to produce the target code for three-address statements	
		TECHNICAL QUESTIONS	
126	Cross Compiler	compiler run on one machine and produce target code for another machine	
		The set of prefixes of right sentential forms	
127	Viable prefixes	that can appear on the stack of a shift- reduce	
127	viable prefixes	parser viable prefixes.	
		The set of items which include the initial item,	
128	Kernel items	$S' \rightarrow .S$, and all items whose dots are not at the	
-		left end	
129	Non kernel items	The set of items, which have their dots at the left	
12)		end	
		It is a process of Compiler which should	
130	Type checking	report an error if an operator applied to an incompatible operand.	
		incompatible operand.	
131	Static checking	the type of variable is known at compile time	
132	Dynamic checking	the type of variable is known at runtime	
133	Input buffering	Technique used to store input string for increasing Compiler speed	
	Deefferrrein	Two buffers are used to store the input string. The	
134	Buffer pair	first buffer and second buffer are scanned	
		alternately.	
135	Sentinels	Special character that is not part of the source	
		program The parse tree containing the values of attributes at	
136	Annotated parse	each node for given input string is called annotated	
	tree	parse tree.	
		A form of recursive-descent parsing that does not	
137	predictive parser	require any back-tracking is known as	
		predictive parser A control flow graph depicts how the	
100		program control is being passed among the blocks.	
138	Control flow	It is a useful tool that helps in optimization by help	
		locating any unwanted loops in the program.	
139	Boolean	Expressions which are composed of the Boolean	
	expression	operators (and, or, and not) applied to elements	
140	Short circuit code	We can also translate a Boolean expression into three-address code without generating	
110		code for any of the Boolean operators	
	1 I		

141	Three address code	It contains three addresses, two for operands and one for the result.		
142	LL grammar	L" stands for left-to-right scanning of the input. "L" stands for constructing a Left most derivation		
143	Stack	Stack is a linear data structure which follows Last in first out (LIFO) order in which the operations are performed		
144	2 pointers in input buffering	Forward pointer Lexeme beginning		
145	Forward pointer		scans ahead until a match for a pattern is found.	
146	Lexeme beginning	points to the beginning of the current lexeme which is yet to be found.		
147	Grouping of phases	Front end Back end	Front end	
148	Usage of sentinel	reduces the two tests to one by extending each buffer half to hold a sentinel character at the end.		
149	Backtracking		if one derivation of a production fails, the syntax analyzer restarts the process using different rules of same production.	
150	Recursive Descent parser		Top-down method of syntax analysis in which a set recursive procedures to process the input is executed	
Facul	ty Team Prepared	T.Manivel,AP/IT	Signature:	

HoD