21BDS0340

Abhinav Dinesh Srivatsa

Compiler Design

Unit – I

I

1. Compiler is a program that converts source program into target program.
2. The output of lexical analyser is tokens.
3. Grammar is a notation to describe regular languages.
4. The Kleen star of a language L is denoted by {}, L, LL, LLL, …
5. NFA allow 0, 1 of more transitions on the same input symbol.
6. Token are identified by the syntax analyser.
7. Parse tree shows the graphical representation of a derivation.
8. Interpreter is a program that reads the source program and executes them line by line
9. Syntax analyser checks whether a token is valid in a language.
10. Code generator is the last phase of a compiler.

II

1. The total number of phases in compiler design is 6.
2. In lexical analysis the original string that comprises the token is called as lexeme.
3. Which of the following translation programs converts assembly language programs to object program? Assembler.
4. The input to preprocessor is a source program.
5. The output of loader link editor is relocatable machine code.
6. The output of compiler is a target assembly program.
7. Compilers are generally written by compiler manufacturer.
8. Pattern is a class defined for similar lexemes of a program.
9. The number of states required to represent the regular expression 0*1 + 0*0 is 3.
10. Regular expression for a language in which the number of a's > number of b's. (aba + a + aab + baa)*.

III

1. Regular expressions are a notation to describe regular language.
2. The smallest part of a program is lexeme.
3. Number of states required to represent 1 + 0*0 is 2.
4. Regular sets are closed under intersection.
5. Object code analysis is not a logical phase of a compiler.

Unit – II

I

1. Predictive parsing is also called as recursive descent parsing.
2. For start symbol S, FOLLOW(S) = $.
3. Non-recursive descent parser is free from ambiguity.
4. In LR parsing, L stands for left.
5. SLR parsing follows canonical parsing technique.

6. SLR, CRR, LALR are types of shift-reduce parsing technique.
7. The most powerful parser is CLR.
8. The relation between SLR, LALR, LR parser is left to right parsing.
9. Recursive descent parser does not follow bottom-up parsing technique.
10. CLR items are also known as core.

II

1. Predictive parser must be free from left recursion.
2. The grammar E -> E+E | E*E | (E) | id is ambiguous.
3. Given S -> xABC, A -> A | ε, B -> b | ε, C -> c | ε. A, Follow(A) = {a, ε}, {b, ε}.
4. LL(k) denotes scanning the input from left to right, producing a left most derivation and using k symbols for lookahead.
5. A non-LL(1) grammar is converted into LL(1) grammar by left factoring, substitution and left recursion.
6. Given E -> E+T|T, T -> T*F|F, F -> (E) | t. The FIRST(E) and FOLLOW(E) are {(, t}, {), $}.
7. Which of the following is a non-recursive predicate parsing? LL(1) parsing.
8. A left factored grammar is suitable for predictive parsing.
9. Pasting table for LL(1) grammar does not have multidimensional entries.
10. The grammar having a production of form p -> ap is called right-recursive grammar.

III

1. CLR – parse the highest number of grammars.
2. SLR(1) grammar is
3. CLOSURE ({[S' -> S]}) – start state of LR(0) automation.
4. LR-parser configuration – (s0s1…sm, ai…an,$).
5. Viable prefix – prefix of right most sentential form.

Unit – III

I

1. The intermediate code generator phase of the compiler takes the parse tree as input.
2. In a parse tree, the nodes represent operators and children of that nodes represent operands.
3. Two forms of type checking are syntactic and semantic.
4. An attribute grammar is a combination of a grammar and a set of semantic rules.
5. Runtime checking is defined as checking of the target program at run time.
6. Type checking done by a compiler is known as static type checking.
7. The three ways of implementing three-address code are quadruples, triplets, and indirect triplets.
8. Semantic analyser adds semantic rules to productions of context free grammar.
9. Traversal order determines an evaluation order for the attribute instance in a given parse tree.
10. DAG stands for direct acyclic graph.

II

1. An S-attribute grammar is the combination of grammar and a set of semantic rules.
2. The general form of three address statement is r := aop.
3. Semantic analyser is used to add semantic rules to production.

4. L-attributed grammar consists of synthesized and inherited attributes.
5. The nature of attributes of T and L is synthesized, inherited.
6. Inherited attributes are used for keeping track of variable declaration, checking for correct use of L-values, and evaluating arithmetic expressions.
7. For which of the following attributes the semantic actions are evaluated before they are expanded? Inherited attribute.
8. Which phase is an optional phase of a compiler? Intermediate code generator.
9. The semantic rule for the production D -> TL is T.type = L.inherit.
10. In DAG representation of basic blocks, interior nodes are labelled by an operator symbol.

III

1. Type synthesis and type inference are two forms of type checking.
2. An attributed grammar that has both synthesized and inherited attribute is L-attribute grammar.
3. Annotated parse tree helps in determining evaluation order for the attribute instance in a parse tree.
4. a = b op c – three address code.
5. In triple notation assignment of location to temporaries is done during code generation.

Unit – IV

I

1. Static allocation is known as compile time allocation.
2. Stack is a block of memory used for managing information needed by a single execution of a procedure.
3. In heap allocation, heap is used to manage dynamic memory allocation.
4. Procedures are implemented by calling sequences, which consists of code that allocates an activation record on the stack.
5. Heap allocation can become stack allocation by using relative addresses for storage in activation records.
6. The basic blocks become the nodes of a DAG whose edges indicate which blocks can follow which other blocks.
7. Register allocation decides what values in a program should be stored in registers.
8. A register descriptor keeps the information about each register.
9. The dynamic programming algorithm partitions the problem of generating optimal code for an express into sub problems of generating optimal code for the sub expressions of the given expression.
10. Code generator is the final phase of a compiler.

II

1. The following storage allocation strategy manages storage for all the data object at compile time. Heap allocation.
2. The allocation that is required for languages that support dynamic data structure – heap.
3. Which of the following is not a field of an activation record? Data size.
4. Static allocation strategy allocates memory at compile time.
5. A pointer called access link is added to activation record.
6. Allocation and deallocation operations are performed by memory manager.
7. Register allocation is the process of deciding which IR values to keep in the registers.

8. Instruction selection is the process of choosing target language instructions for each IR statement.
9. In call by value method, the caller evaluates the actual parameter and pass their values their called procedures.
10. The realloc function causes resizing of allocated memory.

III

1. Data size is not a field of activation.
2. The output of code generator – object code.
3. Automatic variables are also called local variables.
4. An address register is also known as variable descriptor.
5. A data structure used to implement transformation on basic blocks – DAG.

Unit – V

I

1. An occurrence of an express E is called a constant if E was previously computed and the values of the variables in E have not changed since the previous computation.
2. A variable is initialised at a point in a program if its value can be used subsequently.
3. A variable x is said to be a dependent variable if there is a positive or negative constant 'c' such that each time x is assigned its value is increased by 'c'.
4. Algorithms refers to a body of techniques that derive information about the flow of data along program execution
5. The definitions that may reach a program point along some paths are known as declarations.
6. A lattice has a top element denoted T such that for all x in S, x ^ T = x.
7. Optimisers replaces expressions that evaluate to the same constant every time they are executed by that constant.
8. The node 'd' of a flow graph is a parent to node n, if every path from the entry node of the flow graph to n goes through d.
9. In depth first spanning tree, first goes from a node m to a proper descendent of m.
10. A forward edge is an edge a -> b whose head b dominates its tail a.

II

1. Loop optimization improves the performance of CPU.
2. The process of replacing expensive operations by equivalent cheaper operations by target machine is referred to as strength reduction.
3. Which of the following techniques can be used for loop optimisation? Code motion, strength reduction, and induction variables.
4. The piece of code which is not reachable in the program is dead code.
5. The use of the variable x is replaced by the variable yin the subsequent expression in code propagation method.
6. If the expression 3x3.14 is replaced by 9.42 at compile time, then the process is referred to as code motion.
7. Loops are a major part of code optimisation.
8. Global subexpression elimination is an important optimisation technique for pipelined, superscalar and VLIW architecture.
9. a := b + c, d := c, e = b + d – 3 is an example if copy propagation.

10. Common subexpression can also be identified using associative law.

III

1. The transformation of replacing an expensive operation by a cheaper one – strength reduction.
2. Constant propagation – forward data-flow problem.
3. An expression is available at program point P if it has been anticipated along all path reaching P – live-variable analysis.
4. A flow graph is called reducible if its retreating edges in DFST are back edges.
5. The value of ‘x’ as point P could be used along some path in the flow graph starting at P – availability.