Tribhuvan University
Institute of Science and Technology
2078

Bachelor Level / Science	Full Marks: 60 + 20 + 20
Bsc. CSIT	
Compiler Design and Construction	Pass Marks: 24 + 8 + 8
	Time: 3 Hours

Candidates are required to give their answers in their own words as far as practicable.

The figures in the margin indicate full marks.

Section A

Attempt all questions.

1. What are the task performed in lexical analysis. Define DFA. Given regular expression:

   (a+b)*a(a+b) ✓

What are the task performed in lexical analysis. Define DFA. Given regular expression:

(a+b)*a(a+b)

Expert approved answer

The task performed by lexical analysis are:

1.Tokenization: Lexical analysis takes the source code as input and identifies the individual components or tokens that make up the code. Tokens can be keywords, identifiers, literals, operators, or punctuation marks.

2. Filtering: It removes all white spaces and comments and other irrelevant characters that do not contribute to the meaning of program.

3

tab

comments and other irrelevant characters that do not contribute to the meaning of program.

3. It identifies and fills tokens into the symbol table.

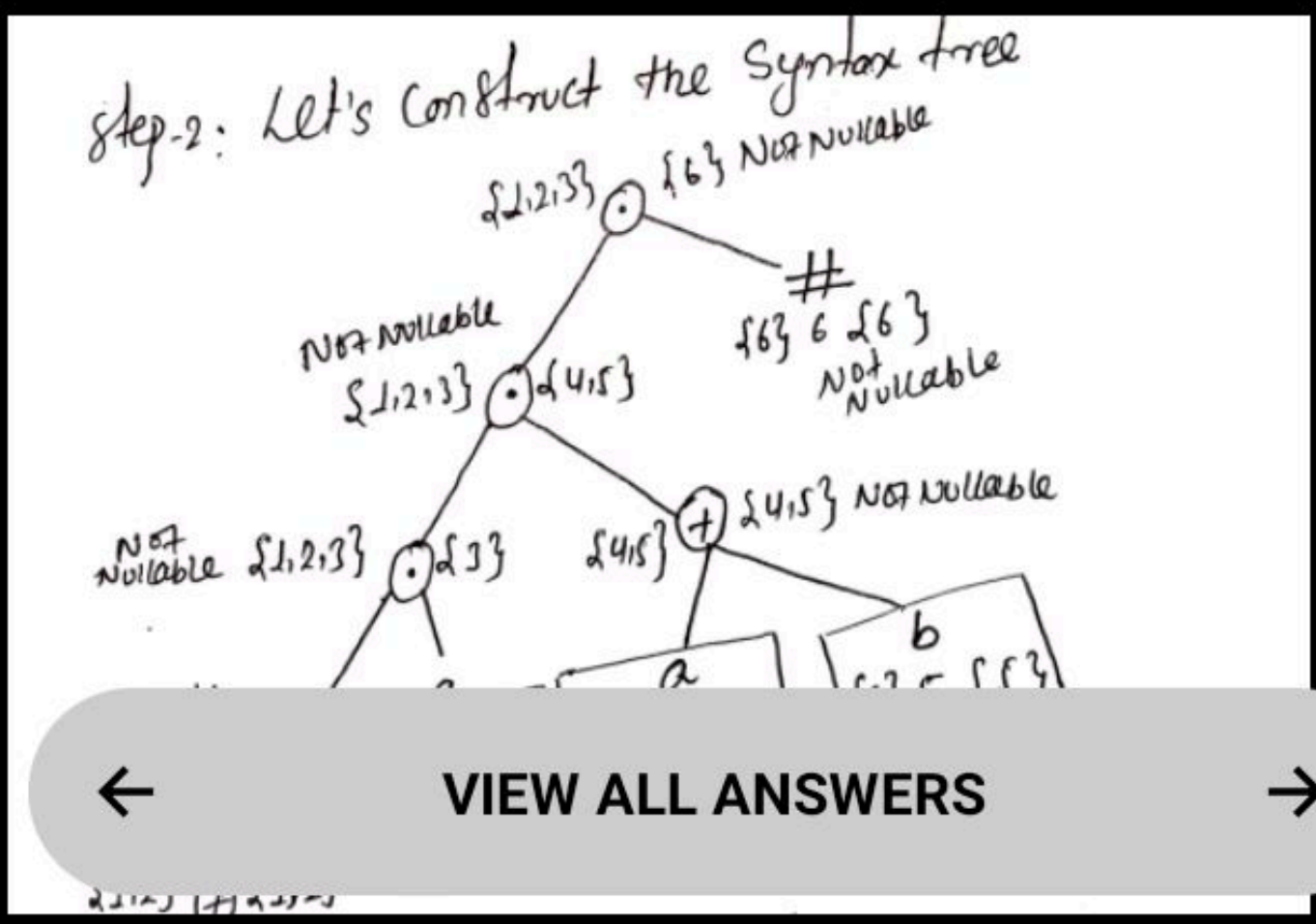
DFA is defined as a finite automata consisting of 5 tuples. It stands for Deterministic Finite Automata.

Given regular expression:

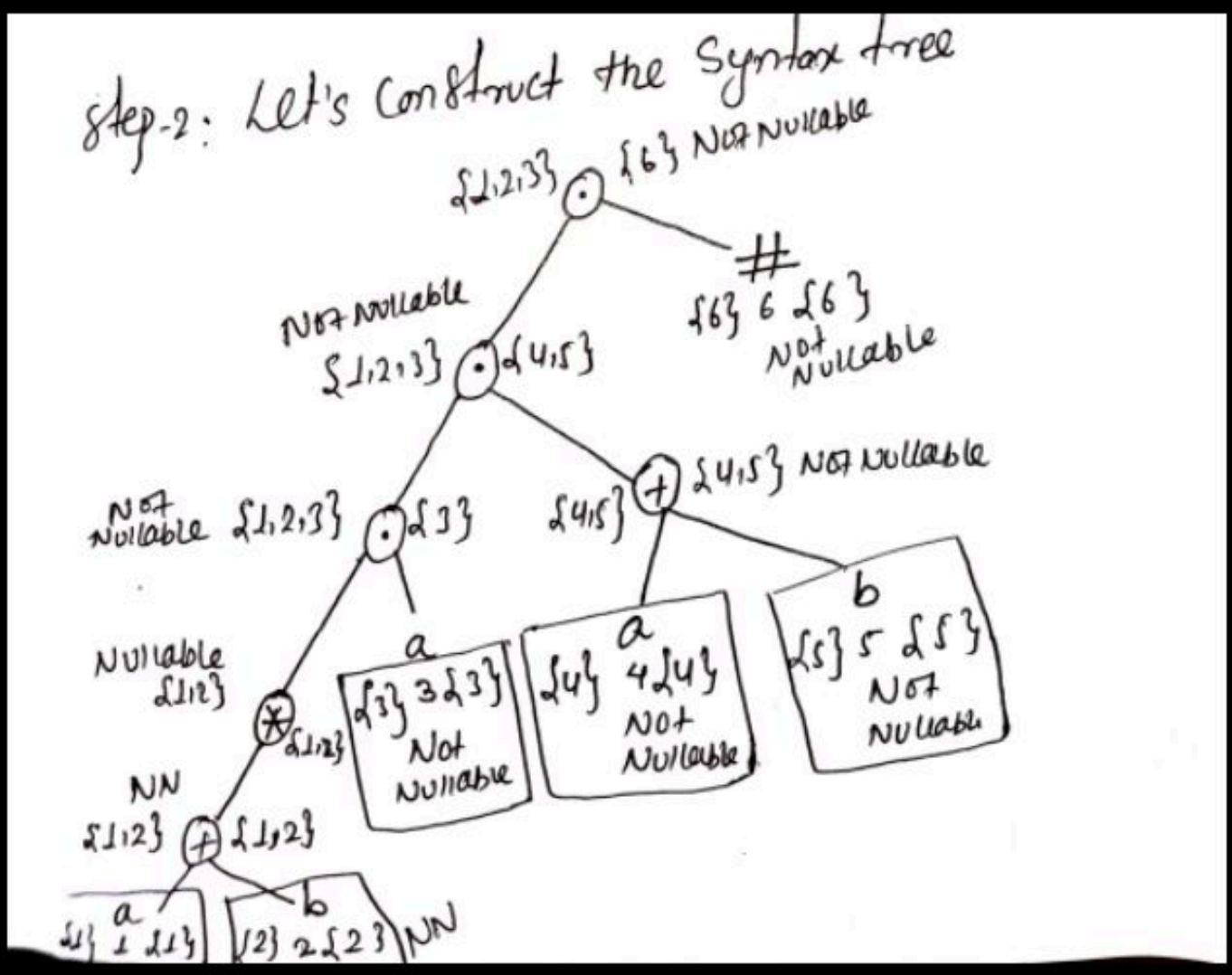
(a+b)*a(a+b)

step 1: Augment the given expression with #

(a+b)*.a.(a+b).#



(a+b)*.a.(a+b).#



Followpos(1)={1,2,3}

Followpos(2)={1,2,3}

Followpos(3)={4,5}

Followpos(4)={6}

Followpos(5)={6}

Followpos(6)={Φ}

Followpos(4)={6}

Followpos(5)={6}

Followpos(6)={Φ}

step 3: Let’s start constructing DFA

start state of DFA= firstpos (root)={1,2,3}=s1

Mark s1:

for a: followpos(1) U followpos(3) = {1,2,3,4,5}=s2

for b: followpos(2) ={1,2,3}=s1

Mark s2:

for a: followpos(1) U followpos(3) U followpos(4) ={1,2,3,4,5,6}=s3

for b: followpos(2) U followpos(5) = {1,2,3,6}=s4

M

{1,2,3,6}=s4

Mark s3:

for a: followpos(1) U followps(3) U followpos(4)=s3

for b: followpos(2) U followpos(5) =s4

Mark s4:

for a: followpos(1) U followps(3) =s2

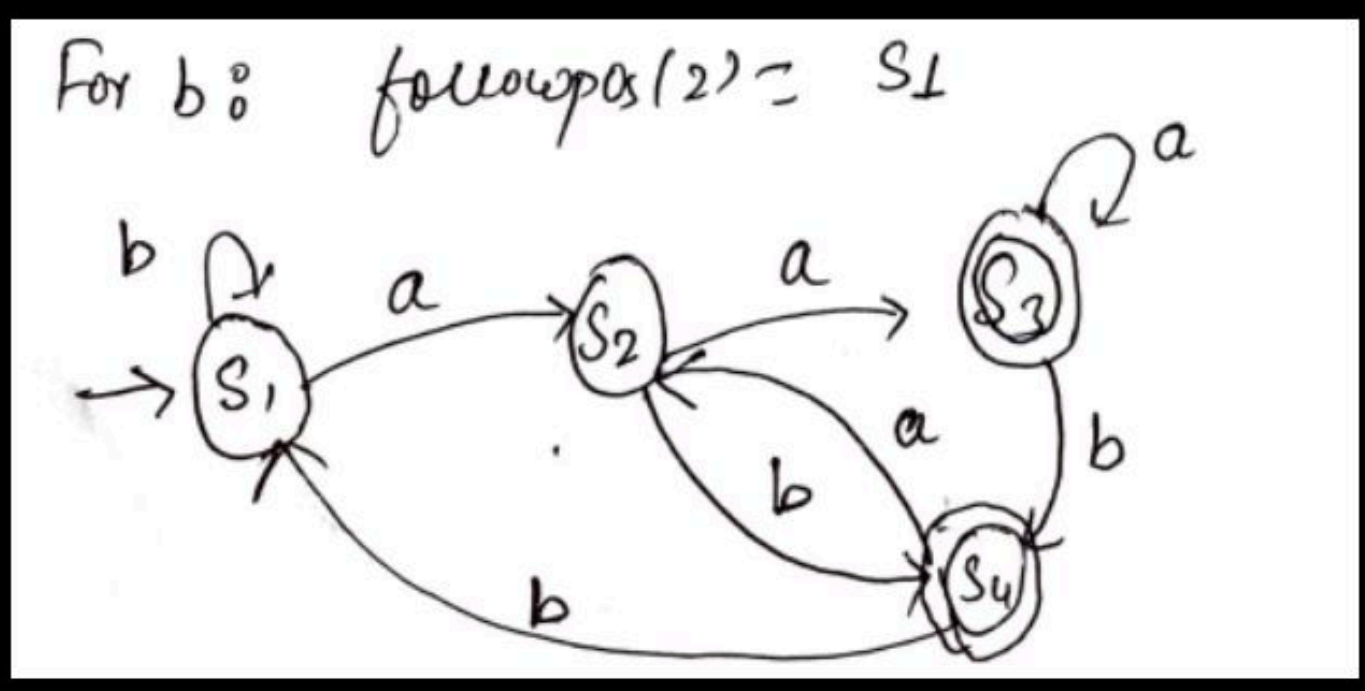


Fig: DFA

QN Bank Solution

Best Answer of the Question

Difference between LR(0) and LR(1) algorithm.
Construct LR(1) parse table for s->AA ,A->aA/b

Expert approved answer

LR(0) Algorithm	LR(1) Algorithm
It does not consider any lookahead symbols.	It is consider one lookahead symbol.
It uses LR(0) items (production rules with a dot).	It uses LR(1) items(production rules with dot and a lookahead symbol).
It is more likely to have shift-reduce conflicts.	Fewer shift reduce and reduce-reduce conflict.
Parsing table has fewer entries.	Parsing table has more entries.

Given grammar

s->AA

A

Given grammar

s->AA

A->aA/b

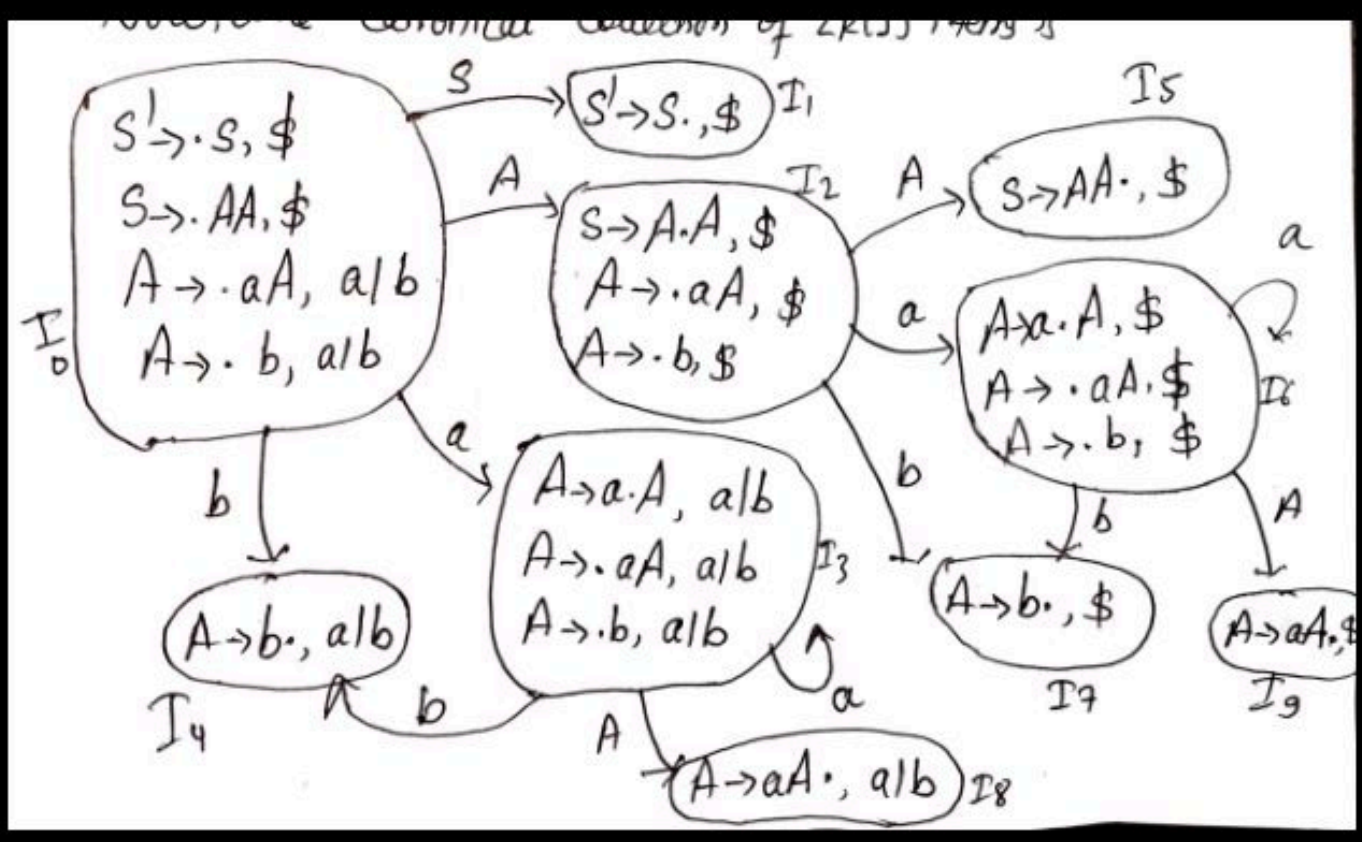
The augmented grammar is

s'->s

s->AA

A->aA/b

Now the canonical collection of LR(1) items is



Now, let's construct LR(1) parse table

	Action			Go to	
states	a	b	$	S	A
0	s3	s4		1	2
1			Accept		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

Since, there are no conflicts, so it is LR(1) parsable.

Type checking is the process of verifying that the types of expressions and variables used in a program are consistent and adhere to languages type system rules. The primary goal of type checking is to identify and prevent type-related errors before the program is executed.

Expert approved answer

Type casting	Type conversion(coercion)
In type casting, a data type is converted into another data type by a programmer using casting operator.	In type conversion, a data type is converted into another data type by a compiler.
Type casting can be applied to compatible data types as well as incompatible	Type conversion can only be applies to compatible data types.

data types as well as incompatible data types.	compatible data types.
In type casting, casting operator is needed in order to cast the data type to another data type.	In type conversion there is no need for a casting operator.
In type casting, the destination data type may be smaller than the source data type, when converting the data type to another data type.	In type conversion, the destination data type can’t be smaller than source data type.
Type casting takes place during the program design by programmer.	Type conversion is done at the compile time.

SDD to carry out type checking:

during the program design by programmer.	done at the compile time.

SDD to carry out type checking:

E->n/E*E/E==E/E[E]?E↑

E->n {E.type=lookup(n.entry)}

E->E_1*E_2 {E.type=(E1.type==E2.type)?
E_1.type:type error}

E->E_1==E_2 {E.type=(E_1.type==E_2.type)? boolean
: type error}

E->E_1[E_2] {E.type=(E_1.type==”array” and
E_2.type==”integer”)? integer: type_error}

E->E_1↑ {E.type=(E_1.type==pointer(t))? t:
type_error}

Difference between compiler and interpreter.

Expert approved answer

	compiler	Interpreter
Definition	It is a program that translates the entire source code of a program into machine code.	It directly executes the source code of a program line by line.
Execution	compilation is done before	Interpretation is done during the execution of the

	code.	
Execution	compilation is done before the program is executed.	Interpretation is done during the execution of the program.
Speed	The compiled code is generally faster as it is optimized for the specific target platform.	The interpreted code is generally slower as the interpretor has to execute the source code line by line.
Error	Compiler reports all the error.	Reports error one at a time.
Memory space	Requires more memory to store generated machine code.	Requires less memory as they don’t generate machine code.

What are the typical entries made in symbol table? Explain.

Expert approved answer

The following are the typical entries made in symbol table:

1. Name:

->Name of identifier

->May be stored directly or as a pointer to another character string.

2.Type

->Type of identifier: variable, label, procedure name

->For variables its type: basic types, derived types

3

->Type of identifier: variable, label, procedure name

->For variables its type: basic types, derived types

3.Location:

->Offset within the program where the current definition is valid.

4.other attributes:

->array limits, fields of records, parameters, return values.

5.scope:

->Region of the program where the current definition is valid.

QN Bank Solution

Best Answer of the Question

What are the disadvantages of shift reduce parsin perform shift reduce parsing of string

w=(x-x)-(x/x) for grammar

E=E-E/ E/E / (E) / x

Expert approved answer

SR parsing is a bottom-up parsing technique. It’s disadvantages are:

-> They have a limited lookaheads.

->They need to perform backtracking.

Given string:

(x-x)-(x/x)

Grammar:

E=E-E/ E/E / (E) / x

QN Bank Solution

Best Answer of the Question

Grammar:

E=E-E/ E/E / (E) / x

stack	Input	Production
$	(x-x)⁻ (x/x)	shift(
$(	x-x)⁻ (x/x)	shift x
$(x	-x)⁻ (x/x)	Reduce E->x
$(E	-x)⁻ (x/x)	shift⁻
$(E⁻	x)⁻ (x/x)	shift x
$(E⁻ X	)⁻(x/x)	Reduce E->x
$(E⁻ E	)⁻(x/x)	Reduce E->E ⁻ E
$(E	)⁻(x/x)	shift)
$(E)	⁻(x/x)	Reduce e-> (E)
$E	⁻(x/x)	shift ⁻
$E⁻	(x/x)	shift (

$E	¯(x/x)	shift ¯
$E¯	(x/x)	shift (
$E¯(	x/x)	shift x
$E¯(x	/x)	Reduce E->x
$E¯(E	/x)	shift)
$E¯(E/	x)	shift x
$E¯(E/x	)	Reduce E->x
$E¯(E/E	)	Reduce E->E/E
$E¯(E	)	shift)
$E¯(E)	$	Reduce E->(E)
$E¯E	$	Reduce E->E ¯ E
$E	$	Accept

VIEW ALL ANSWERS

QN Bank Solution

Best Answer of the Question

Define attribute grammar with example of inherited and synthesized attributes

Expert approved answer

Attribute grammar is a special form of context free grammar where some additional information (attribute) are append to one or more of its non terminals in order to provide context sensitive information. Each attribute has a well defined domain of values , such as integer, float, character, string and expressions.

Ex:

E->E+T {E.valve= E.valve + T.valve}

If the valve of attribute depends only upon it’s children then it is synthesized attribute.

Ex:

h

child nodes {A.b.C} then it is said to be a

Ex:

E->E+T {E.valve= E.valve + T.valve}

If the valve of attribute depends only upon it’s children then it is synthesized attribute.

Ex:

here, S->ABC, if S is taking valves from its child nodes (A,B,C) then it is said to be a synthesized attribute.

If the valve of attribute depends on the parent or siblings then it is called inherited attribute.

Ex:

S-> ABC, if A gets valve from S, B, C or if B gets valve from S, A, C, likewise C gets valve from S, A, B then it is called inherited attribute.

Define three address code. Write down Quadruples for a=-b*(c+d)/e

Expert approved answer

The address code that uses at most three addresses for operands and one for result is called three address code. Each instruction in 3AC can be described as 4-tuples coperator, operand 1, operand 2, result). Ex: x=y+z.

Given :

a=-b*(c+d)/e

Let’s write three address code

t1=-b

t2=c+d

t3=t1*t2

t

t2=c+d

t3=t1*t2

t4=t3/e

a=t4

-> Quadruples

index	operator	arg 1	arg 2	result
(0)	–	b		t1
(1)	+	c	d	t2
(2)	*	t1	t2	t3
(3)	/	t3	e	t4
(4)	=	t4		a

fig: Quadruples for given grammar

List out the different types of runtime storage management techniques.

Expert approved answer

Runtime storage management or dynamic memory management deals with the allocation, deallocation and organization of memory during program execution.

Tow of the most commonly used runtime storage management techniques are:

i) stack allocation

ii) Heap allocation

Stack storage allocation

The allocation of memory during run time using stack is called stack storage allocation. Stack is a Last In First Out(LIFO) storage structure where new storage is allocated and d
the stack.

The allocation of memory during run time using stack is called stack storage allocation. Stack is a Last In First Out(LIFO) storage structure where new storage is allocated and deallocated at only one “end” called the top of the stack.

->Storage is organized as stack and activation records are pushed and popped as activation begin and end, respectively.

->At runtime, activation record can be allocated and deallocated by incrementing and decrementing top of the stack.

Advantages:

->Supports recursion as memory is always allocated on block entry.

->allows creating data structure dynamically.

What are the advantages of code optimization.

Define Dead-code elimination.

Expert approved answer

The advantages of code optimization are:

1. Faster Execution: Optimized code runs faster and performs computations more efficiently, resulting in reduced execution time.

2. Reduced Resource Usage: Optimized code consumes fewer system resources, such as CPU cycles, memory, and disk space. By utilizing system resources efficiently, you can optimize the overall performance of your software and improve the scalability of your application.

3. Improved User Experience: Optimized code leads to a smoother and more responsive user experience.

4. Lower Costs: Optimized code can reduce hardware requirements and save on

5.

4. Lower Costs: Optimized code can reduce hardware requirements and save on infrastructure costs.

5. Extended Battery Life: For software running on battery-powered devices, code optimization can help conserve energy.

6. Easier Maintenance: Well-optimized code tends to be more modular, readable, and organized. This makes it easier for developers to understand, maintain, and modify the codebase over time.

Dead-code elimination:

Dead code elimination is a process in software development where unused or unreachable code is identified and removed from the program. This optimization technique improves the efficiency and readability of the codebase.

Ex:

unoptimized code:

i

VIEW ALL ANSWERS

from the program. This optimization technique improves the efficiency and readability of the codebase.

Ex:

unoptimized code:

```
i=0;

if(i==1)

{

a=x+i;

}
```

Optimized code:

i=0;here, i is already initialized as 0. so there is no need of the part i==1.

Factors affecting (target code generator) code generator/code generator design issues

Expert approved answer

1. Input to the code generator-> The input to the code generator is the intermediate representation together with the information in the symbol table.

2. the target program-> The output of the code generator is the target code. The target code comes in three forms: absolute machine language, relocatable machine language and assembly machine language.

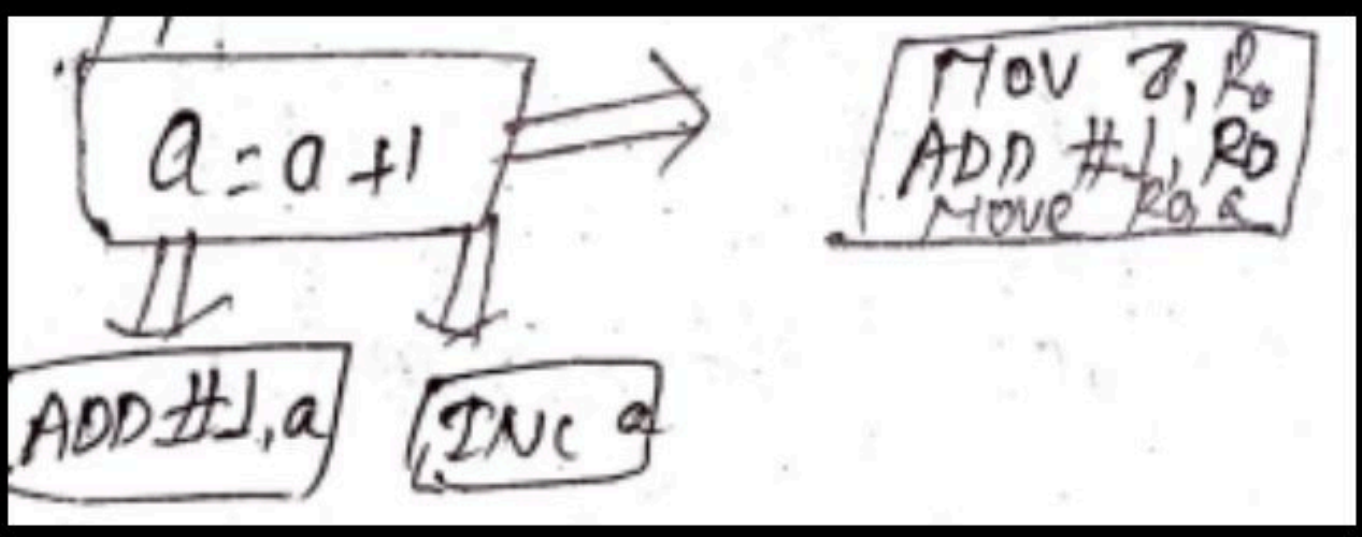
3. The target machine-> Implementing code generation requires understand of the target machine architecture and its instruction set.

4. Instruction selection-> Instruction selection is important to obtain efficient code, suppose w

assembly machine language.

3. The target machine-> Implementing code generation requires understand of the target machine architecture and its instruction set.

4. Instruction selection-> Instruction selection is important to obtain efficient code, suppose we translate 3AC as:



5. Register allocation: Since registers are the fastest memory in the computer, the ideal solution is to store valves in register we must choose which valves are in the register at any given time.