Q1

A

Lexical analyser:

a : identifier

[: left bracket

index : identifier

] : right bracket

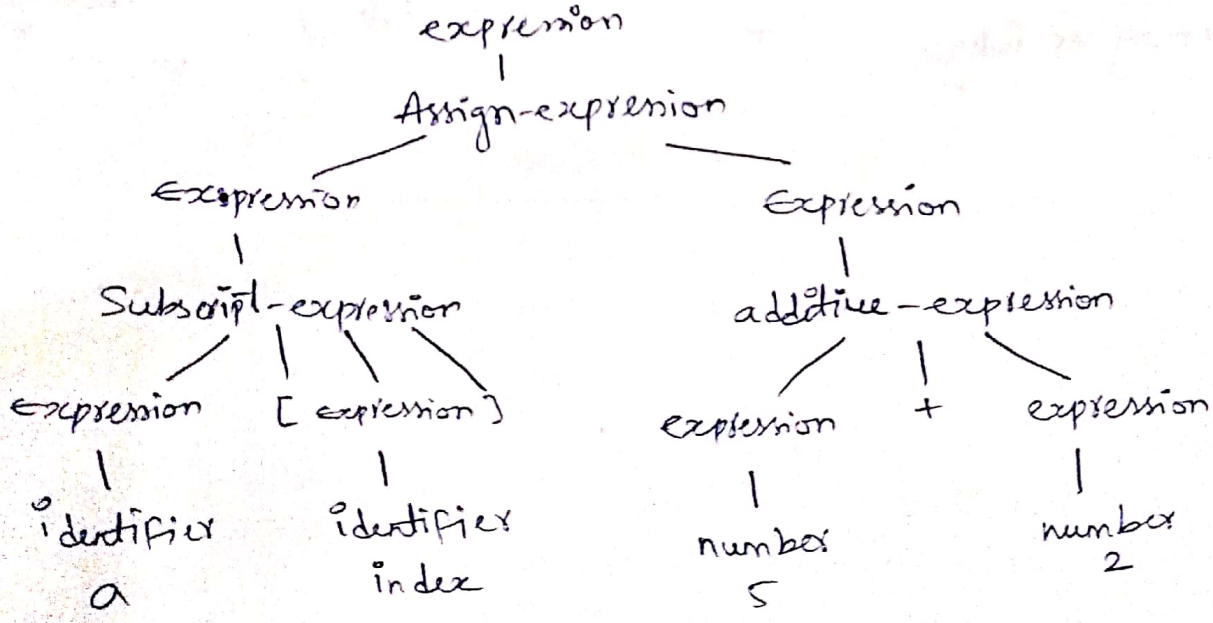
= : assignment

5 : number

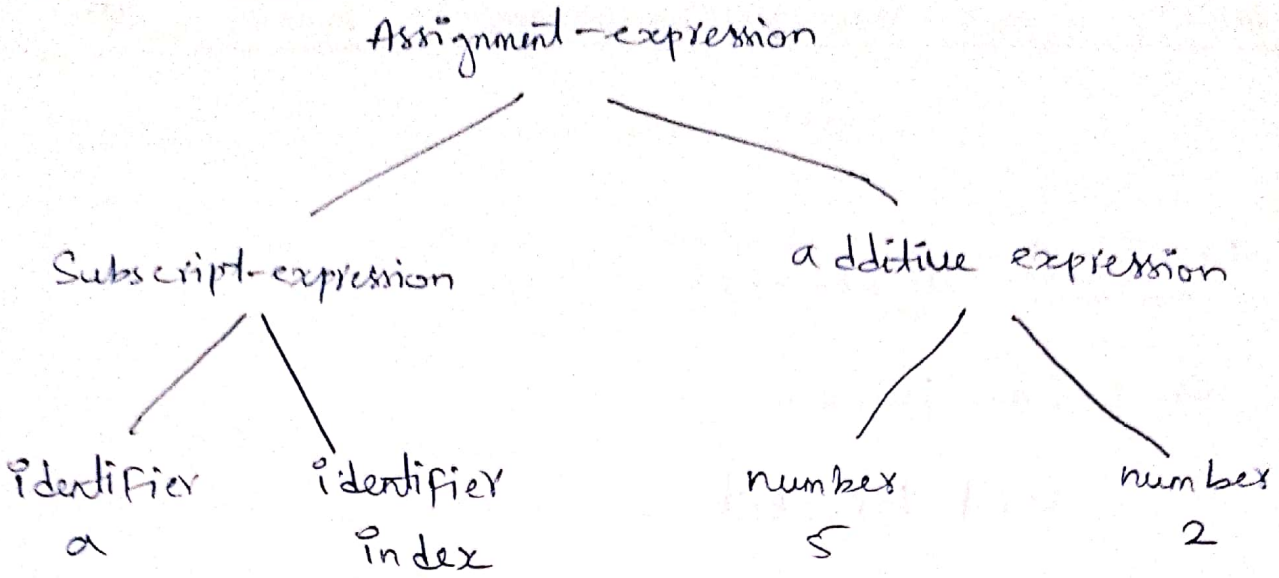
+ : plus sign

2 : number

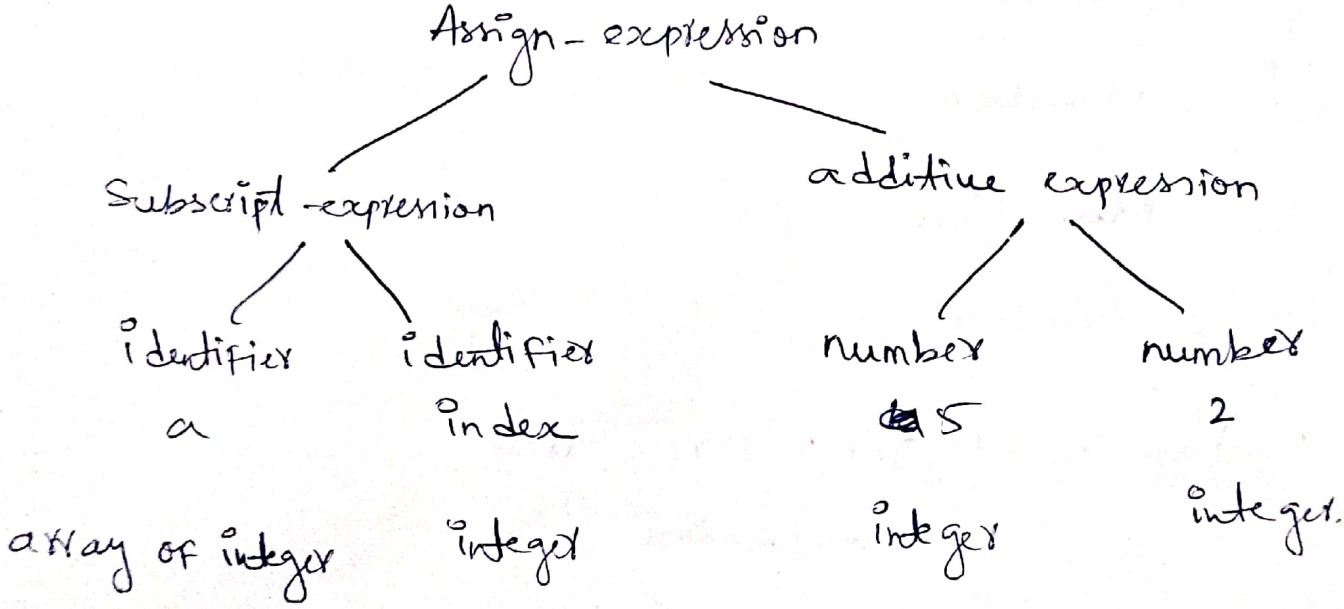
Syntax analyser: Parse tree



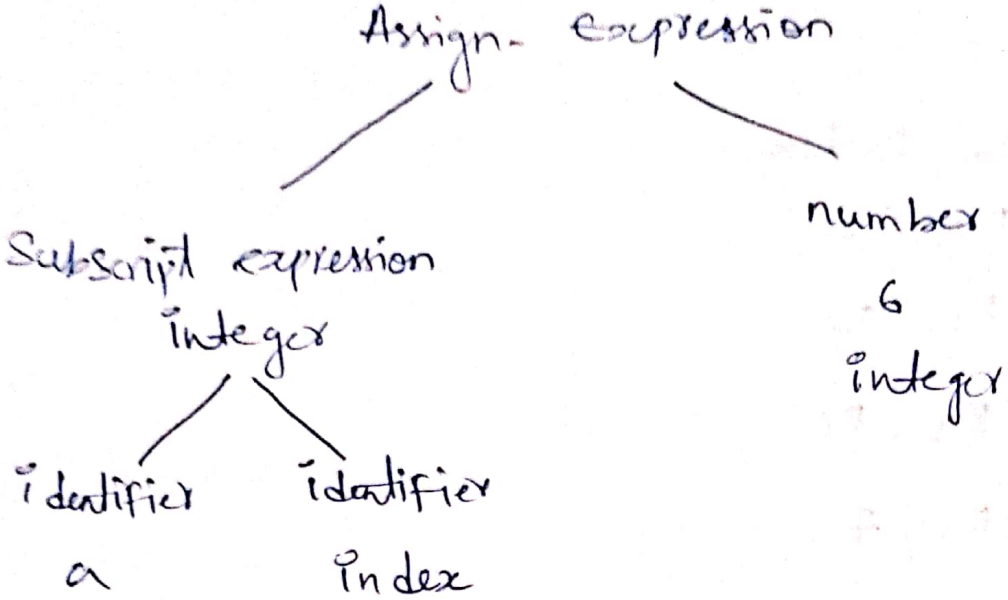
Syntax tree



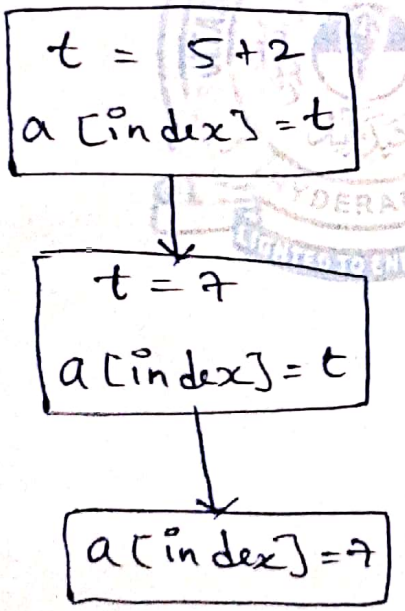
Semantic Analyzer



Optimisation or Annonated Tree



optimization of intermediate code:



Code generation

a [index] = 7

↓

```
MOV   R0, index
MOV   R0, 2
MOV   R1, &a
ADD   R1, R0
MOV   R1, 7
```

Target code Optimizer

```
MOV   R1, index
SHL   R0
MOV   &a[R1], 7
```

Augmented Grammar

$S' \rightarrow .S$

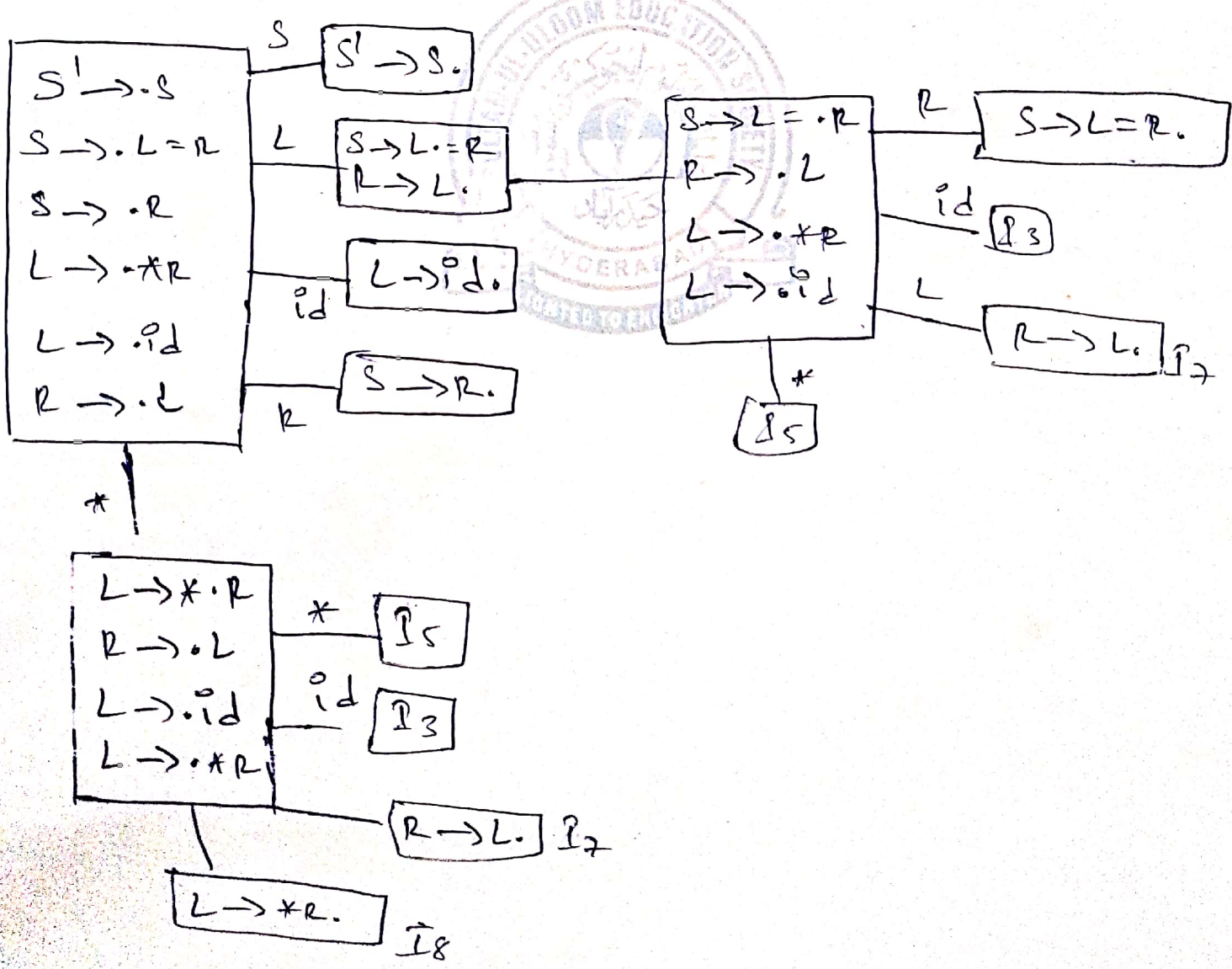
$S \rightarrow .L = R$

$S \rightarrow .R$

$L \rightarrow .*R$

$L \rightarrow .id$

$R \rightarrow .L$



Follow (S') : {$}

Follow (S) : {$}

Follow (L) : {$, =}

Follow (R) : {$, =}

State	Action				goto		
	id	=	*	$	S	L	R
0	s3		s5		1	2	4
1				accept			
2		s6/r(R→L)					
3		r(L→id)		r(L→id)			
4				r(S→R)			
5	s3		s5			7	8
6	s3		s5			7	9
7		r(R→L)		r(R→L)			
8		r(L→R)		r(L→R)			
9				r(S→L=R			

the shift/reduce conflict on state 2 when look ahead is an =

∴ the grammar is not SLR.

Q3 Find out First & Follow for the following grammar

$S \rightarrow ACB \mid Ba \mid Cd$

$A \rightarrow et \mid \epsilon$

$B \rightarrow gh \mid d$

$C \rightarrow d \mid e \mid \epsilon$

First[S] = d e g ε

First[A] = e ε

First[C] = d e ε

First[B] = d g

Follow[S] = {$}

Follow[A] = {d, e, g, $}

Follow[C] = {d, g}

Follow[B] = {e, $}

Q4 Differentiate between static run time environment & stack based run time environment

Static RTE	Stack RTE
1) Does not make data structure & object dynamically	1) Makes data structure & object dynamically
2) Does not support recursive procedure	2) Support recursive procedure
3) At compile time the data object name are fixed	3) index & register points the memory addressing
4) This strategy is easy & simple in implementing	4) Slower than static allocation
5) Allocation of all data objects is performed at compile time	5) allocation of data objects is performed at run time

Production	Semantic's
$L \rightarrow En$	$L.val = E.val$
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
$T \rightarrow f$	$T.val = f.val$
$f \rightarrow (E)$	$f.val = E.val$
$f \rightarrow digit$	$f.val = digit.lexval$

Q6 Translate the assignment statement a[i] = b * c - b * d into quadruples, triples and indirect triples

A) Converting to three address code

$T_1 = b * c$

$T_2 = b * d$

$T_3 = T_1 - T_3$

$a[i] = T_3$

quadruple:

	OP	Arg1	Arg2	Result
(0)	*	b	c	T_1
(1)	*	b	d	T_2
(2)	-	T_1	T_2	T_3
(3)	:=	T_3		a[i]

triple:

	OP	arg1	arg2
(0)	*	b	c
(1)	*	b	d
(2)	-	(0)	(1)
(3)	=	(2)	

Introduct topic:

201 (0)
202 (1)
203 (2)
204 (3)

Q7) Write a short note on Basic block & explain its construction

The basic block is a straight line code sequence with no in & out branches except at the beginning and end. It is a set of statements that always run one after the other sequentially without any halt

A compiler first converts the source code of any programming language into an intermidiate code. It is then converted into basic block. After partitioning an intermidiate co into basic blocks, the flow of control among basic block is represented by a flow graph