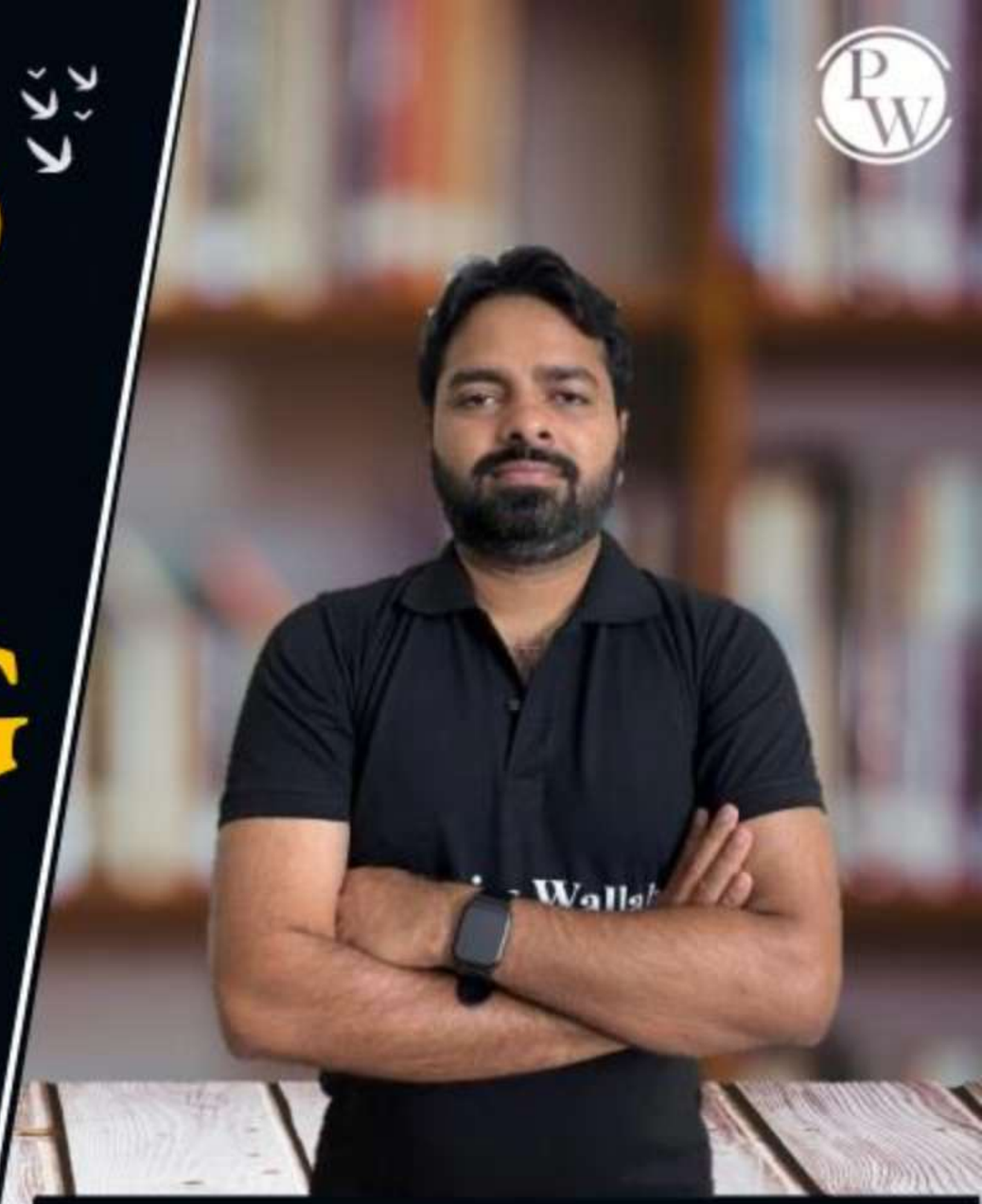
MISSION ISRO CS & IT ENGINEERING

Compiler Design

Lecture No.- 01



By- Mallesham Devasane Sir

Topics to be Covered

Topic

Topic Name

↳ Lexical Analysis

Q1) Which of the following is Translator ?

A) Compiler

B) Interpreter

C) Assembler

D) All of the above

E) None of these

Q2) Which of the following is correct phase of compiler?

A) Lexical

B) Syntax

C) Semantic

D) Code Generation

E) ~~All of these~~

7 phases

↳ Lexical
Syntax
Semantic
ICG
CO
CG
CO

Q3) Which of the following is used in Compiler ?

A) Symbol Table ⇒ In all phases

B) parsing Table ⇒ In syntax Analysis

C) Stack

D) All of the above

E) None of these

Q4) Symbol Table maintains _____ information.

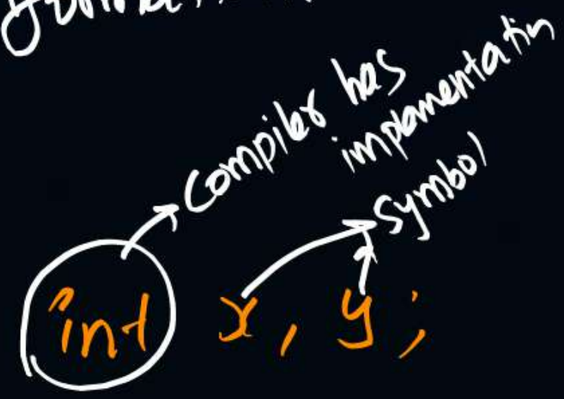
A) Operators

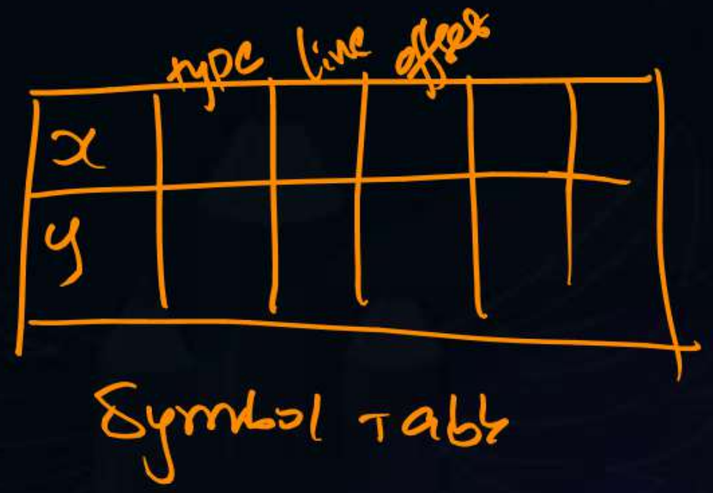
B) Statements

C) Identifiers ✓

D) Keywords

E) All of them





Q5) Relocation does by _____

A) Compiler — H.L.L ⟹ L.L.L

B) Assembler — A.L.L ⟹ M.L.L.

C) Loader — Relocation

D) Linker — Resolves external references

E) All of these

Q6)

```
#define Add(x) x+x

Void main( )
{
    int a, b;
    a = 20;
    b = 30;
    a = Add(b) * a;
    printf("output = %d", a);
}
```

constant

Expression

Replace with exp

$a = b + b * a;$

more precedence

$= 30 + 30 * 20$

$= 630$

7) Find the error type in the following code.

```
int x, y;
x = 10;
y = x * z;
```

z → not declared

A) Lexical
B) Syntax
C) Semantic
D) All of these
E) None of these

8) Find no. of tokens in above code.

A) 13
B) 14
C) 15
D) 16

int	x	y
x	=	=
,	10	x
y	;	*
;		z
		;

9) Which of the following can help to design Lexical Analysis?

All can do

A) Finite Automata → sufficient
B) PDA
C) LBA
D) TM
E) All of these

I

A) FA
B) Regular Exp
C) Regular Grammar
D) All of these
E) None

II

10) Which of the following is sufficient to design lexical Analysis?

A) FA
B) PDA
C) LBA
D) TM

11) Find functionality of lexical Analyzer.

A) Recognizes tokens

B) Ignores comments

C) Reports lexical errors

D) uses longest prefix rule

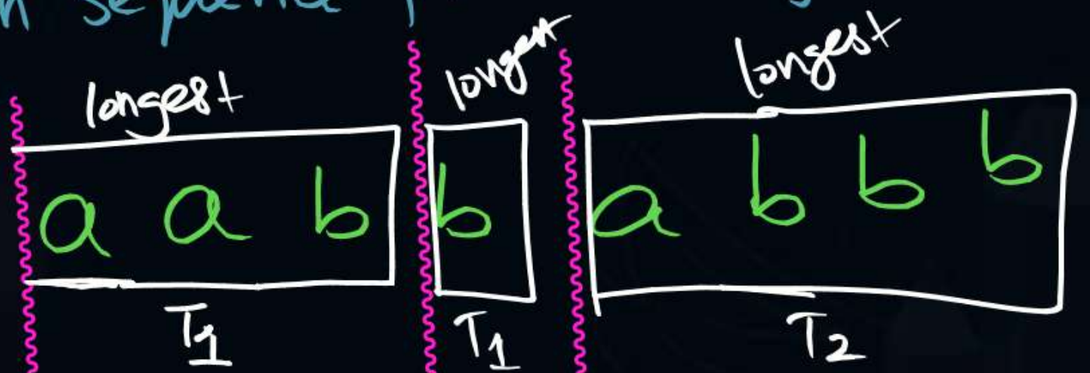
E) All of these

12) $T_1 = a^* b$

$T_2 = a b^*$

Input = aabbabbb

Identify the token sequence produced by lexical analyzer.



$= T_1 T_1 T_2$

13)

```
int 2xy;
```

Find error in above code.

Lexical Error

14)

```
int x2y;
```

Find error.

No error

15) `int xy+2;`

Find error.

No lexical error

Syntax error
expecting = instead of +

16) Match the following groups.

Group - I	Group - II
1. Lexical error ← b/c	a. void main( ) { x=y+z; }
2. Syntax error ← d	b. #include<stdio.h> main( ){ f( ); }
3. Semantic error ← a	c. void main( ){ int x=092; }
4. Linker error ← b	d. void main( ){ int x }

x, y, and z are not declared

092 → not octal digit

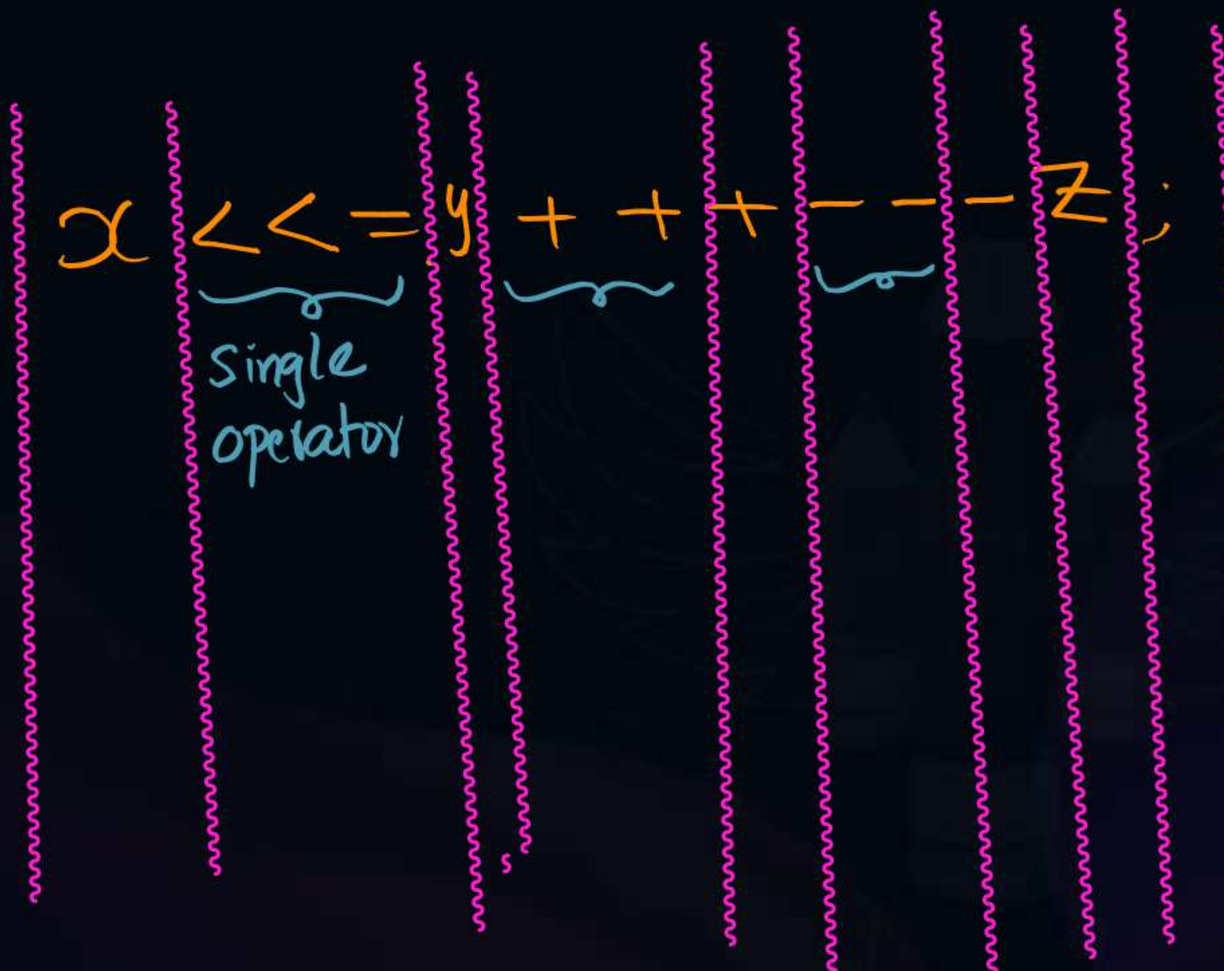
octal

Syntax error

(17) Find no. of tokens.

```
x<<=y+++---z;
```

= 9 tokens



(18)

$T_1 = ab^*c$

$T_2 = a^*bc$

$T_3 = abc^*$

Find TOken sequence.

Q1) Input: aabaabbc

Lexical error

Q2) Input: abbc bc abbc

T_1 T_2 T_1

$\Rightarrow T_1 T_2 T_1$

Q3) Input: abcc aabc ac

T_3 T_2 T_1

$\Rightarrow T_3 T_2 T_1$

(19) Identify Single token.

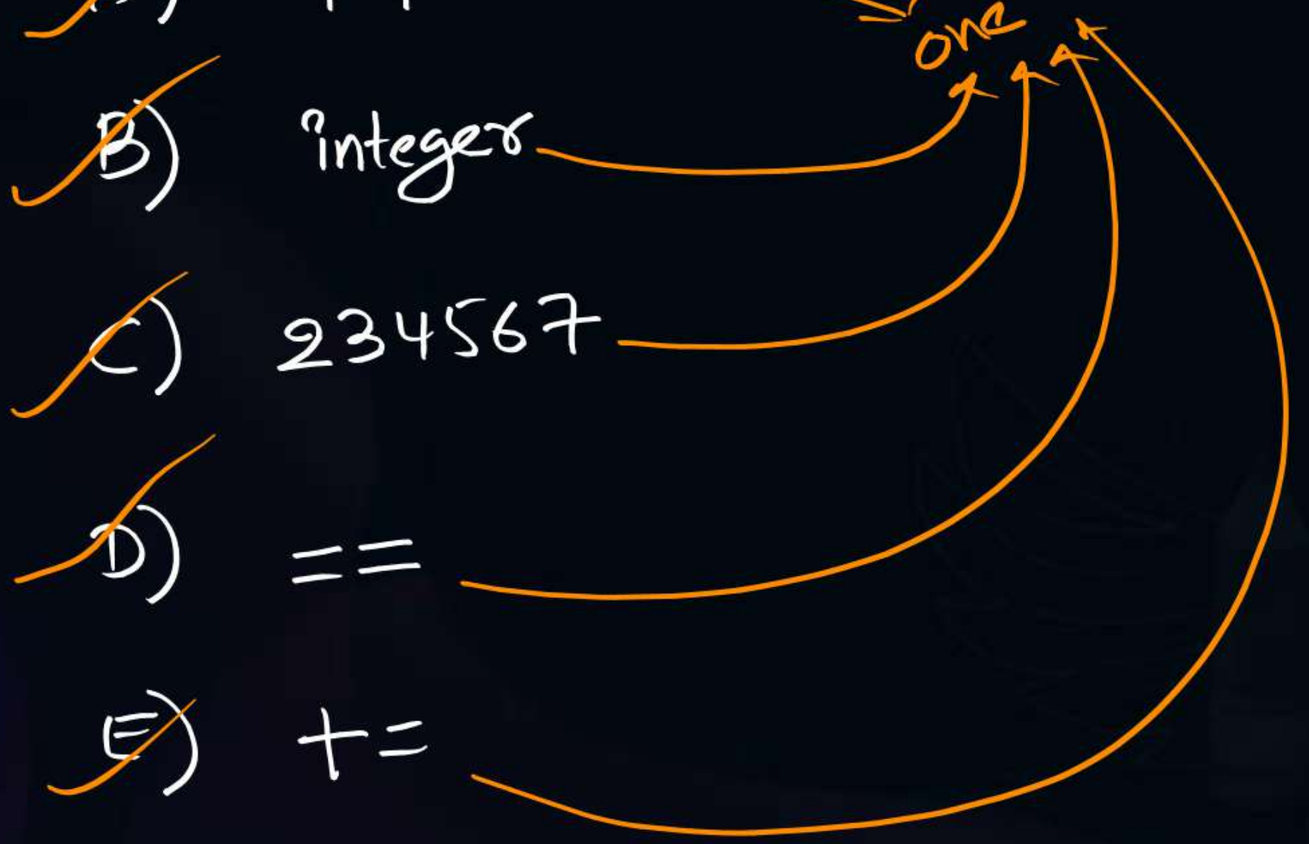
A) ++

B) integer

C) 234567

D) ==

E) +=



(20) Match the following groups.

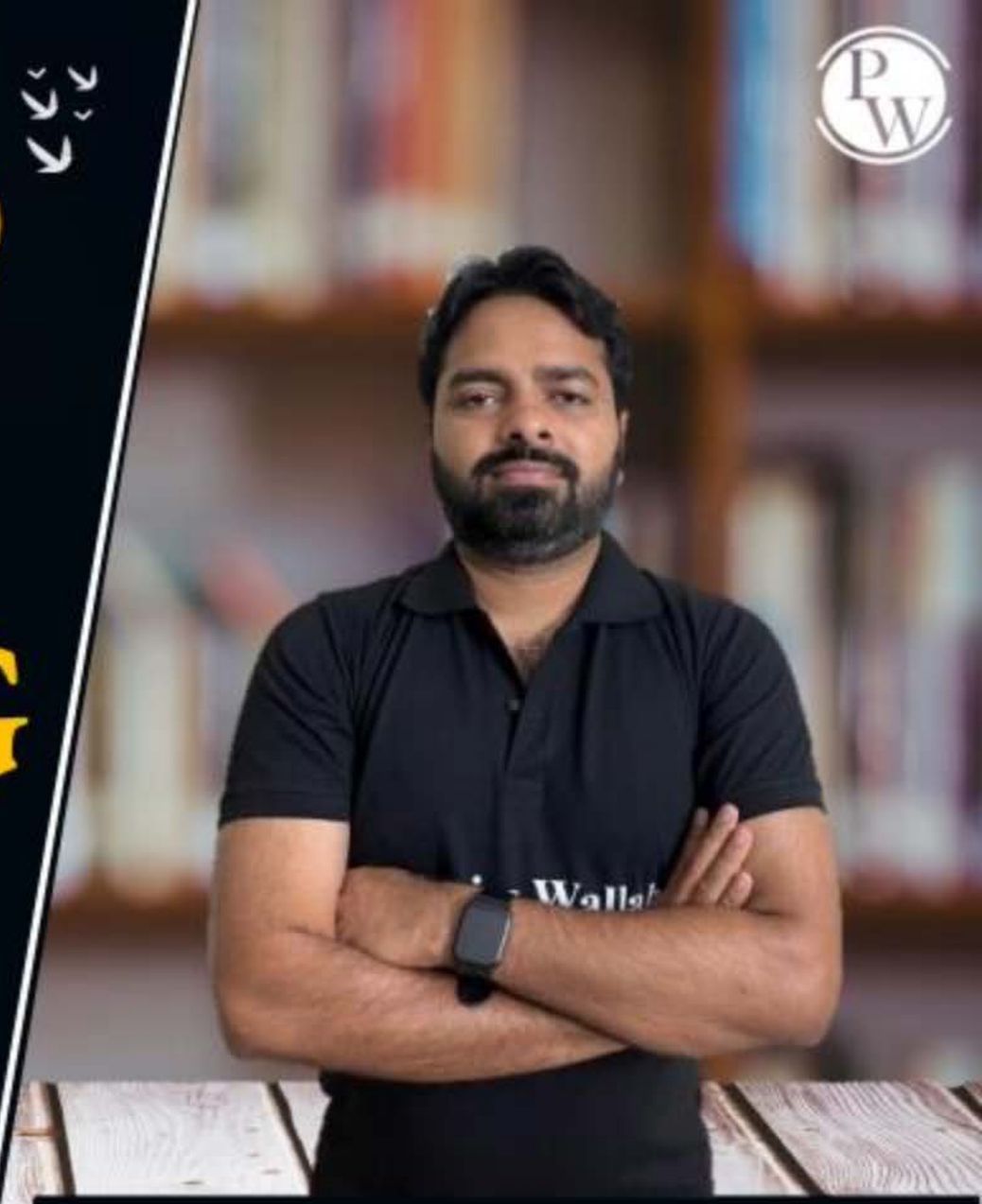
Group I	Group II (output)
1. Lexical phase	a) character stream
2. Syntax phase	b) Token stream
3. Semantic phase	c) parse Tree
4. Intermediate code	d) Annotated parse Tree
	e) Syntax Tree / TAC / DAG / CFG

THANK - YOU

MISSION ISRO CS & IT ENGINEERING

Compiler Design

Lecture No.- 02



By- Mallesham Devasane Sir

Topics to be Covered

Topic

Syntax Analysis

- CFG
- Top-down parser
- Bottom up parser

(1)

```
void main()
{
    int x, y;
    x = y+ ;
}
```

A) Lexical Error

B) Syntax Error

C) Semantic Error

D) None

②
```
int x;
x = y+z;
```
} Semantic Error
y and z are not declared

③ `x=y;` } semantic error

④ `x y;` } Semantic error

⑤
```
int x,y;
x=y;
```
} No compile time error

⑥
```
typedef int x;
x y;
```
} No error

(7) $S \to SaS \mid a$

$a \Rightarrow 1$ PT

$aaa \Rightarrow 1$ PT

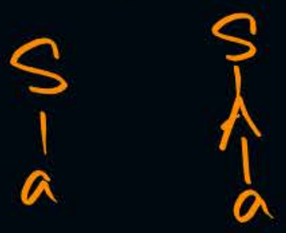
$a^5 \Rightarrow > 1$ PT

Ambiguous

(8) $S \to a \mid A$

$A \to a \mid b$

$a \Rightarrow > 1$ PT



Ambiguous

(9) $E \to E+E \mid E*E \mid a$

Ambiguous

(10) $E \to E+a \mid a$

$a \Rightarrow 1$ PT

$a+a \Rightarrow 1$ PT

$a+a+a \Rightarrow 1$ PT ...

Unambiguous

Ambiguous CFG?
Unambiguous CFG?

Ambiguous
↳ $\exists w$, > 1 PT

Unambiguous
↳ $\forall w$, 1 PT

Slide 5

(11) $S \to AB \mid ab$

$A \to axy$ ✓

$B \to f$ ✓

CFG is ___

A) Ambiguous CFG

B) LL(1) CFG

C) Left Recursion

D) Not LL(1) ✓

$First(AB) \cap First(ab)$

$\{a\} \cap \{a\} \neq \phi$

$\{ab, axy\}$

---

(12) $S \to SS \mid a$

CFG is ___

A) Amb. CFG ✓

B) LL(1)

C) Left Recursive CFG ✓

D) Not LL(1)

LL(1) CFG

↳ It is unambiguous CFG
It is not having Left Recursion
It is Left factored.

$S \to ab \mid ac$

common prefix

(Non Left factored)

	a
S	$S \to ab$ $S \to ac$

$\Rightarrow$

$S \to aX$

$X \to b/c$

Left factored

(13)

$S \rightarrow AbB \mid f$

$A \rightarrow gA \mid h$

$B \rightarrow aA \mid e$

First(S) = {g, h}

First(A) = {g, h}

First(B) = {a, e}

Follow(S) = {$}

Follow(A) = {b, $}

Follow(B) = {$}

gA h

$First(S) = Fi(AbB) \cup Fi(f)$

$= \{g, h\} \cup \{f\}$

$= \{f, g, h\}$

(14)

$S \to ABC \mid$

$A \to ab \mid \varepsilon$

$B \to fgh \mid \varepsilon$

$C \to bad \mid \varepsilon$

LL(0) ?
LL(1) ?
LL(2) ?
LL(3) ?
LL(4) ?

$Fi(S) = \{a, f, b, \varepsilon\}$

$Fi(A) = \{a, \varepsilon\}$

$Fi(B) = \{f, \varepsilon\}$

$Fi(C) = \{b, \varepsilon\}$

$Fo(S) = \{\$\}$

$Fo(A) = \{f\} \cup \{b\} \cup Fi(S)$
$= \{f, b, \$\}$

$Fo(B) = \{b, \$\}$

$Fo(C) = \{\$\}$

(15) $S \to a$

LL(0) ✓
LL(1) ✓
LL(2) ✓

(16) $S \to a \mid b$

LL(0) ✗
LL(1) ✓

(17) $S \to ab \mid ac \mid abd$

LL(0) ✗
LL(1) ✗
LL(2) ✗
LL(3) ✓

(18) $S \to ab \mid abc \mid ade$

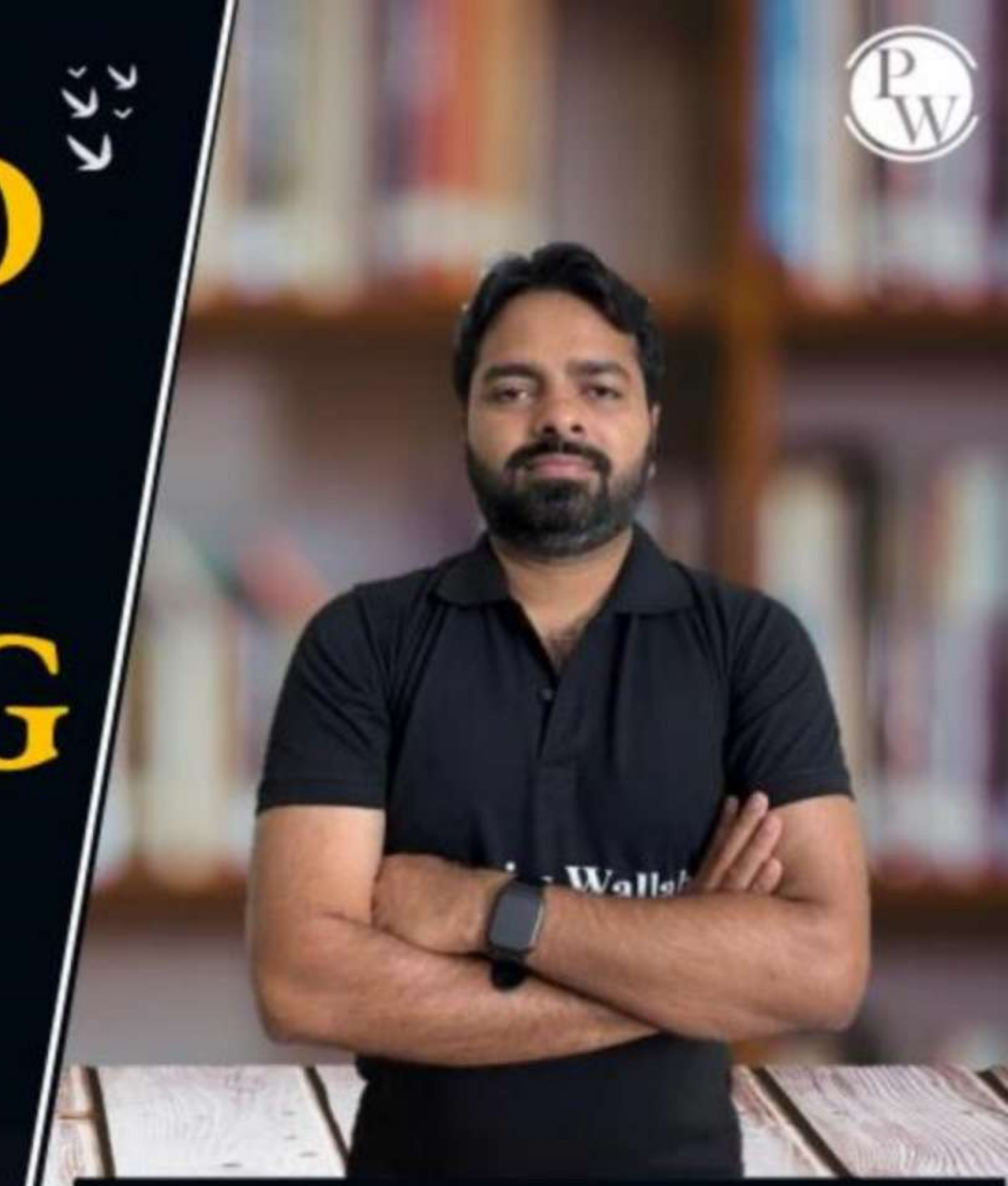
LL(3) ✓

THANK - YOU

MISSION ISRO CS & IT ENGINEERING

Compiler Design

Lecture No.- 03



By- Mallesham Devasane Sir

Topics to be Covered

Topic

Syntax Analysis

- CFG
- Top-down parser
- Bottom up parser

CFG	Ambiguous CFG	Left Recursive CFG	Non Left factored [common prefixes]	LL(1) CFG
$V \to any$	$\exists w$, >1 PT	$S \to Sa \mid b$	$S \to ab \mid acd$	$S \to a \mid bc$
Example:	$S \to a \mid \varepsilon \mid Aa$			
$S \to SS \mid aS \mid \varepsilon$	$A \to \varepsilon$			
	$\Downarrow$			
	$a \Rightarrow$ 2 PTs (>1 PT)			

LL(1) CFG	LR(0)	SLR	LALR	CLR LR(1)
LL(1) Table ↳ no entry has more than 1 production	LR(0) Table ↳ No conflict present	SLR(1) Table ↳ no conflict	No conflict in LALR(1) Table	no conflict in CLR(1) Table

① If $X \to \alpha . t\beta$ is shift item and $Y \to \alpha .$ is reduced item then which of the following is **SR** Conflict in LR(0) ?

A) $X \to \alpha . t\beta$ and $Y \to \alpha . t\beta$ are present in same state.

B) $X \to \alpha . t\beta$ (Shift) and $Y \to \alpha .$ (Reduce) are present in same state

C) $X \to \alpha .$ and $Y \to \alpha .$ are present in same state

D) None of these

(2) If $X \to \alpha.t\beta$ is shift item and $Y \to \alpha.$ is reduced item then which of the following is RR Conflict in LR(0) ?

A) $X \to \alpha.t\beta$ and $Y \to \alpha.t\beta$ are present in same state.

B) $X \to \alpha.t\beta$ (Shift) and $Y \to \alpha.$ (Reduce) are present in same state

C) $X \to \alpha.$ (R) and $Y \to \alpha.$ (R) are present in same state

D) None of these

(3) If $X \to \alpha . t\beta$ is shift item and $Y \to \alpha .$ is reduced item then which of the following is SR Conflict in SLR ?

A) Both Shift item and Reduced item are present in same state

B) Both shift and Reduced item are present but terminal of Shift item present in follow of reduced item.

C) Both A & B

D) None

LR(0) | SLR | CLR & LALR

SR conflict:

$X \to \alpha . t\beta$

$Y \to \alpha .$

$\vdots$

If $t \in Follow(Y)$

$X \to \alpha . t\beta, L_1$

$Y \to \alpha ., L_2$

If $t \in L_2$

RR conflict:

$X \to \alpha .$

$Y \to \alpha .$

$\vdots$

$Follow(X) \cap Follow(Y)$ is not empty

$X \to \alpha ., L_1$

$Y \to \alpha ., L_2$

$L_1 \cap L_2$ is not empty

(4) Which of the following is correct order of expressive power of parsers?

1. $LL(1) < LR(0) < SLR$
2. $LR(0) < SLR < LALR < LL(1) < LR(1)$
3. $LL(0) < LL(1) < LL(2)$
4. $LR(0) < LR(1) < LR(2)$
5. $LL(1) < LR(1)$
6. $LR(0) < SLR < LALR < CLR$
7. $LL(1) < LALR$
8. $LL(1) < SLR$
9. $LR(1) < CLR$

(5) Identify LL(1) CFG.

A) $S \to a$

B) $S \to Sa \mid b$

C) $S \to aS \mid b$

D) $S \to aSb \mid b$

E) $S \to aSa \mid \varepsilon$

6) Which of the following is sufficient to prove CFG is LL(1)?

A) CFG is Unambiguous, non Left Recursive, and Left factored

B) LL(1) Table has maximum 1 production in each entry

C) Both A and B

D) None

$S \rightarrow aSa \mid \epsilon$

	a
S	$S \rightarrow aSa$ $S \rightarrow \epsilon$

7) $S \rightarrow aSa \mid \varepsilon$

Above CFG is ____

A) Unambiguous CFG

B) Non Left Recursive

C) Left factored

D) LL(1)

E) Not LL(1)

8) $S \rightarrow SS \mid a$

A) Ambiguous CFG

B) LL(1)

C) LR(0)

D) CLR

E) neither LL(1) nor LR(1)

(9) Ambiguous CFG is _____

A) Never be LL(1)

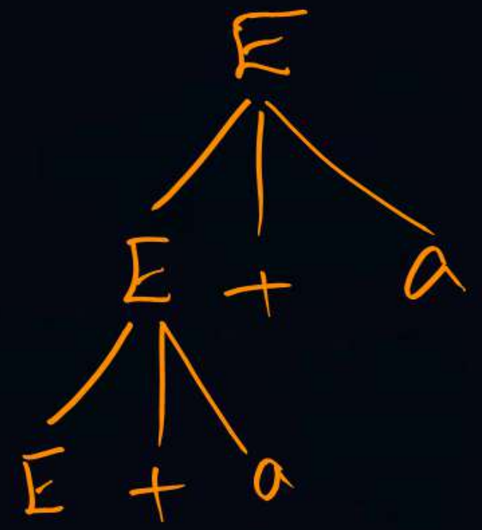
B) " " LR(0)

C) " " SLR

D) " " LALR

E) " " CLR

10) $E \rightarrow E + a \mid a$



A) + is Left Associate

B) + is Right "

C) + has no associative

D) None

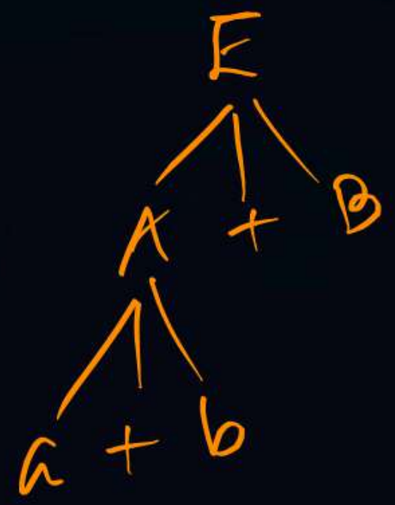
11) $E \rightarrow a+E \mid a$

+ is Right Associative

12) $E \rightarrow A+B$

$A \rightarrow a+b$

$B \rightarrow c$



+ is Left Associative

13) What is Inherited Attribute ?

A) Computation depends on children

B) " " " parent and/or siblings

C) computation depends on parent/siblings/children.

D) None of these

14) What is Synthesized Attribute ?

A) Computation depends on children

B) " " " parent and/or siblings

C) computation depends on parent/siblings/children.

D) None of these

15) Match the following Groups.

Group - I	Group - II
A) Inherited Attribute	I) $A \rightarrow BC \{A.x = B.x + C.x\}$
B) Synthesized Attribute	II) $A \rightarrow B \{A.x = B.x + C.x\}$ C (A.x: Depends on children)
C) L-attributed Grammar	III) $A \rightarrow BC \{B.x = A.x\}$
D) S-attributed Grammar	IV) $A \rightarrow BC \{C.x = B.x + A.y\}$

A → III, IV
B → I, II
C → I, II, III, IV
D → I

L-attributed Grammar

↳ computation depends on parent / Left siblings / children

$A \rightarrow BC \{ A.x = B.y \}$

$A \rightarrow BC \{ B.x = A.z \}$

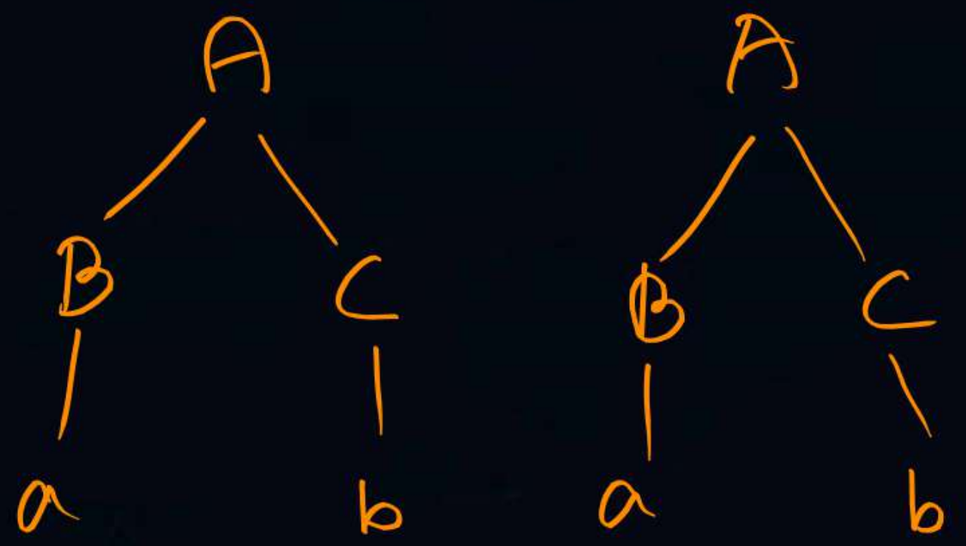
~~$A \rightarrow BC \{ B.x = C.x \}$~~

16) $A \rightarrow BC$ {print 1}

$B \rightarrow a$ {print 2}

$C \rightarrow b$ {print 3}

Find o/p for input = "ab".



Give SDT is

S-attributed grammar ⇒ Reverse of RMD (Bottom up)

Non terminals order

L-attributed grammar ⇒ All translations from Left to Right

Translations Order

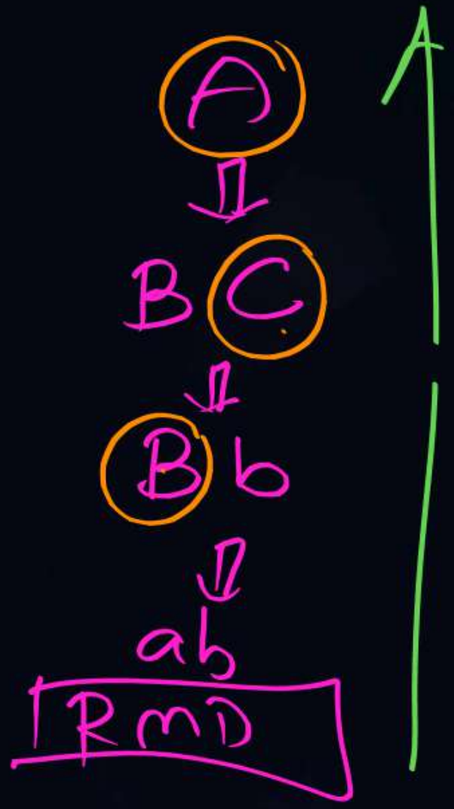
16)

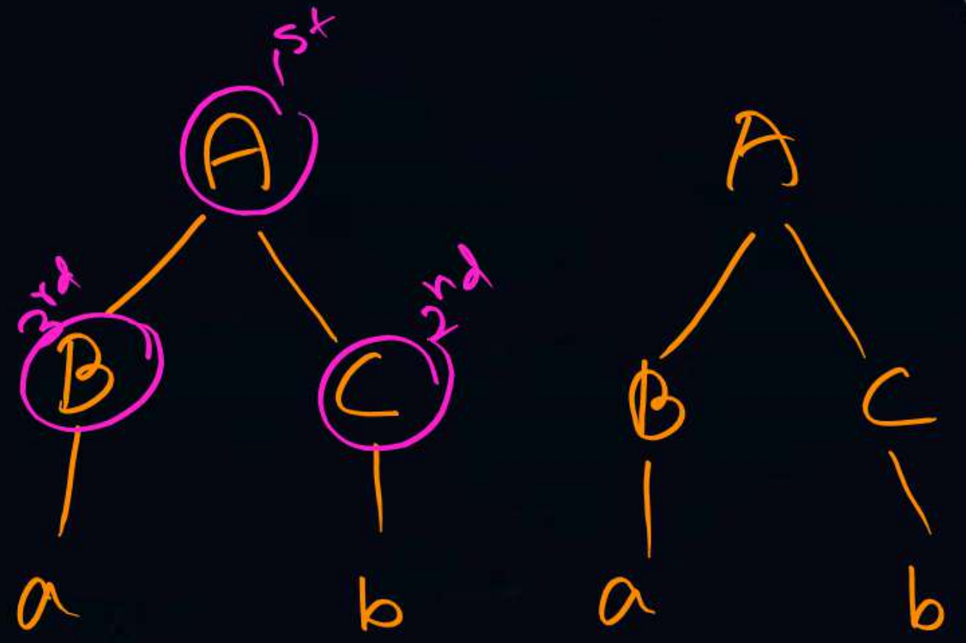
$A \rightarrow BC$ {print 1}

$B \rightarrow a$ {print 2}

$C \rightarrow b$ {print 3}

Bottom-up parsing:





RMD Numbering: 1st, 2nd, 3rd

Reverse of RMD: 3rd, 2nd, 1st

Op: 2 3 1

16) $A \rightarrow BC$ {print 1}

$B \rightarrow a$ {print 2}

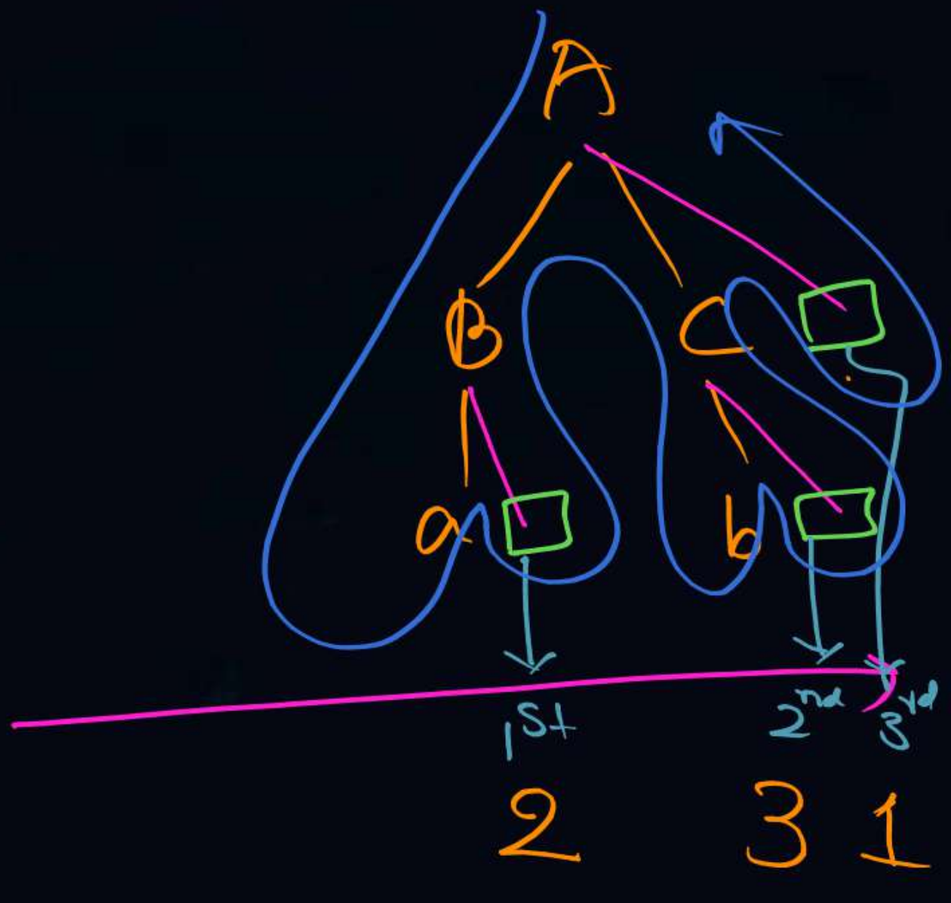
$C \rightarrow b$ {print 3}

L-attributed Evaluation:

Topological order

Inorder

Depth left, Left to right

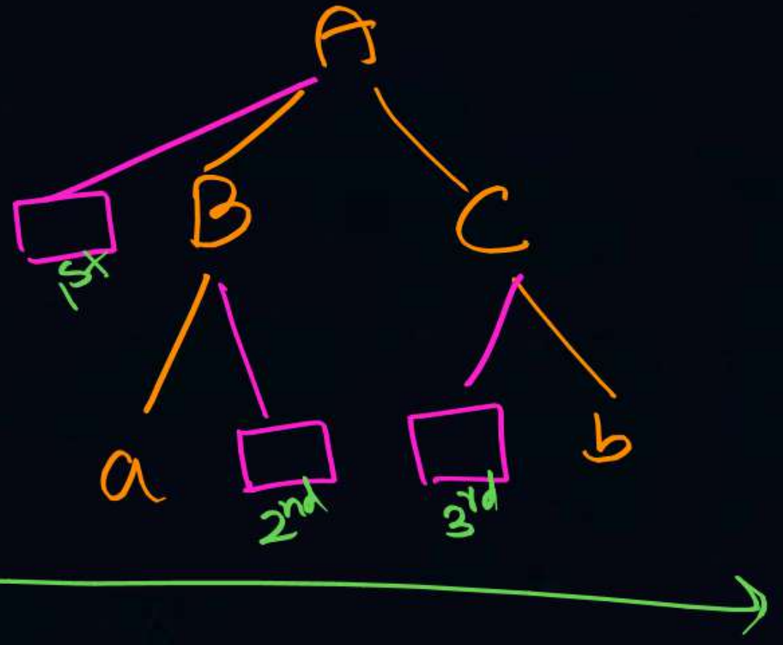


(17)

$A \rightarrow \{print\ `00'\}\ BC$

$B \rightarrow a\ \{point\ `55'\}$

$C \rightarrow \{point\ `77'\}\ b$



Input : ab

O/p = ?

Q1) No evaluation technique mentioned

O/p : 005577

Translations order from L to R :

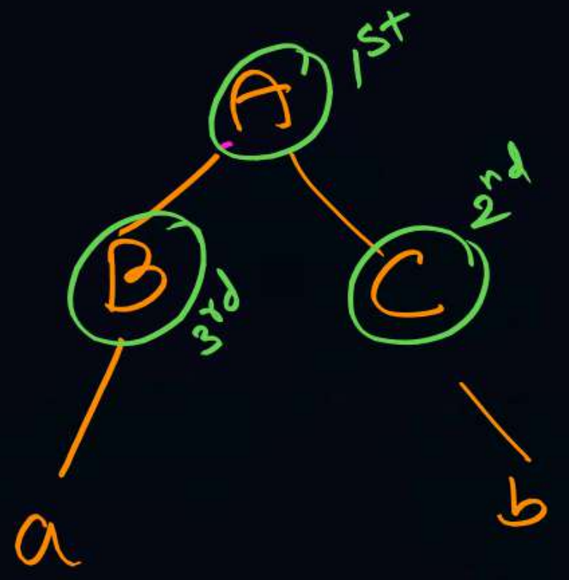
1st	2nd	3rd
00	55	77

(17)

$A \rightarrow \{print\ `00'\}\ BC$

$B \rightarrow a\ \{print\ `55'\}$

$C \rightarrow \{print\ `77'\}\ b$



Input : ab

O/p = ?

Q2) Use Bottom up parsing.
(LR parsing)
(SR parsing)
(Reverse of RMD)

RMD Non-terminal ordering:
1st 2nd 3rd

O/p: 00 77 55

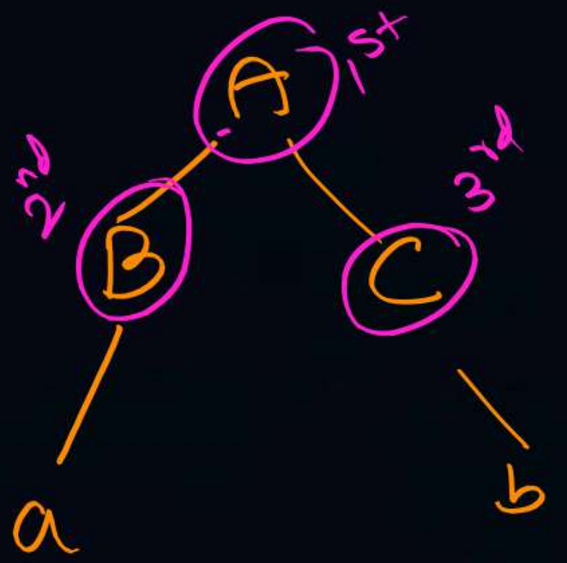
Bottom up parsing: 557700

(17)

$A \rightarrow \{print\ `00'\}\ BC$

$B \rightarrow a\ \{print\ `55'\}$

$C \rightarrow \{print\ `77'\}\ b$



Input : ab

O/p = ?

Q3) use TOP down parsing
(LMD)
(LL(1))
(predictive parser)

LMD non-terminal numbering:
1st 2nd 3rd

O/p : 00 55 77

THANK - YOU

MISSION ISRO CS & IT ENGINEERING

Compiler Design

Lecture No.- 04

By- Mallesham Devasane Sir

Topics to be Covered

Topic

Intermediate code

Code optimization

Syntax Analysis

① What are Intermediate Representations ?

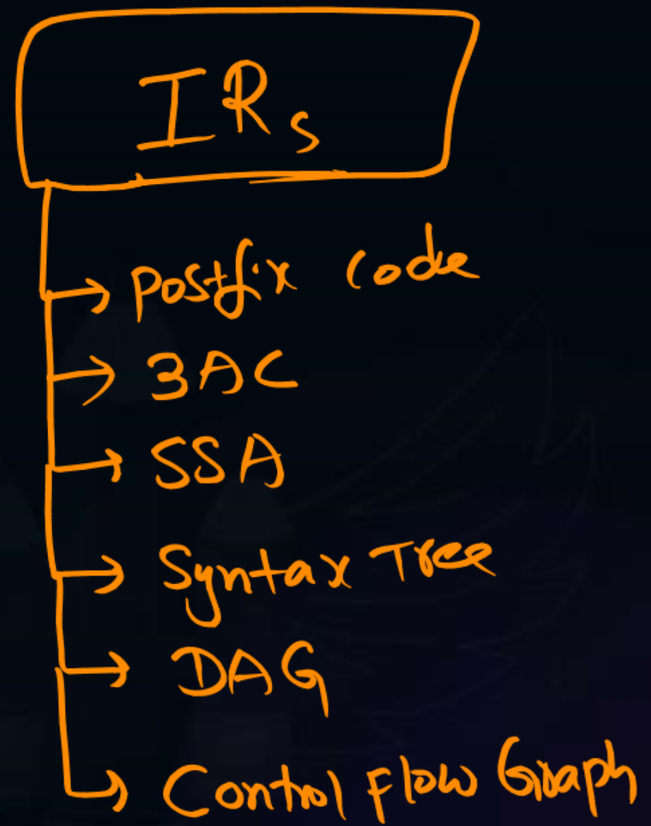
A) Syntax Tree

B) Directed Acyclic Graph

C) Three Address Code

D) Control Flow Graph

E) All of the above ✓



(2) Convert the following expression into 3AC.

$$x = a + a + a + a$$

(I)

$t_1 = a + a$

$t_2 = a + a$

$x = t_1 + t_2$

4 variables

(II)

$t_1 = a + a$

$t_2 = t_1 + a$

$t_3 = t_2 + a$

$x = t_3$

5 variables

(III)

$t_1 = a + a$

$x = t_1 + t_1$

3 variables

(IV)

$a = a + a$

$x = a + a$

2 variables

(V)

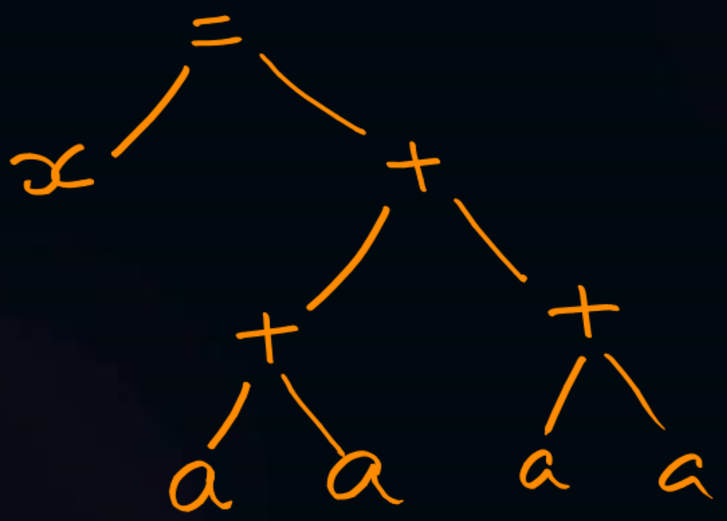
$a = a + a$

$a = a + a$

1 variable

Best 3AC

③ What is Syntax Tree for $x = (a+a)+(a+a)$



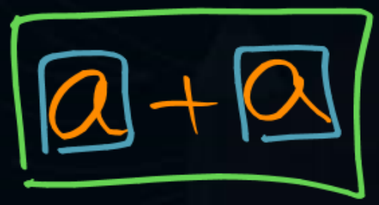
Leaf node → operand

Non leaf → operator

④ What is DAG for $x = (a+a)+(a+a)$

3 nodes
4 edges

DAG
↳ To eliminate Common Subexpressions

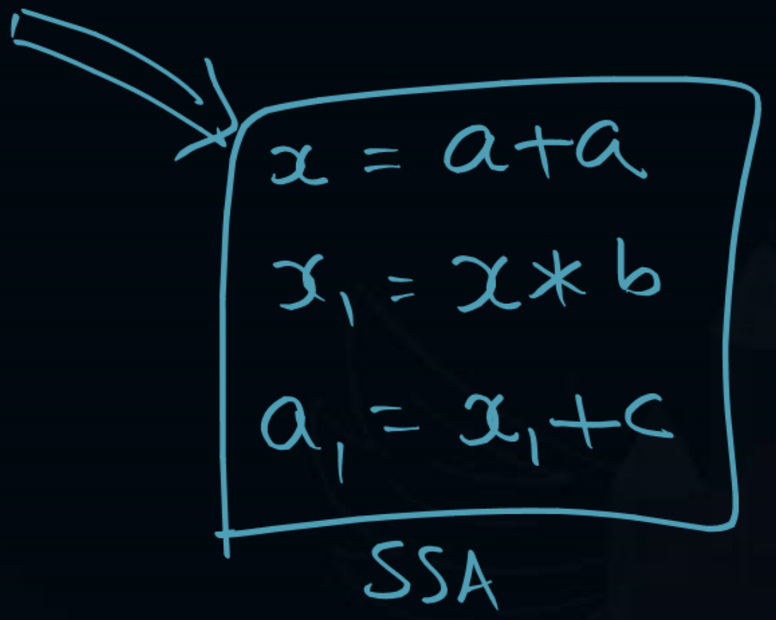


(5) Convert the following 3AC into SSA.

$x = a + a$

$x = x * b$

$a = x + c$



6 variables

a
b
c
x
x_1
a_1

SSA code
↳ It is also 3AC but every variable has single definition

⑥ Write Triple form and Quadruple form for the following 3AC.

```
a = a + b
b = a * c
c = b + d
```

Triple Form

XXX : (+, a, b)

YYY : (*, XXX, c)

ZZZ : (+, YYY, d)

Quadruple

AAA : (+, a, b, a)

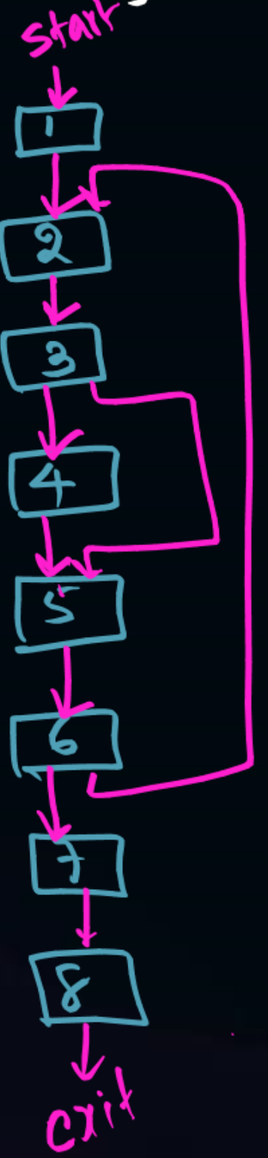
BBB : (*, a, c, b)

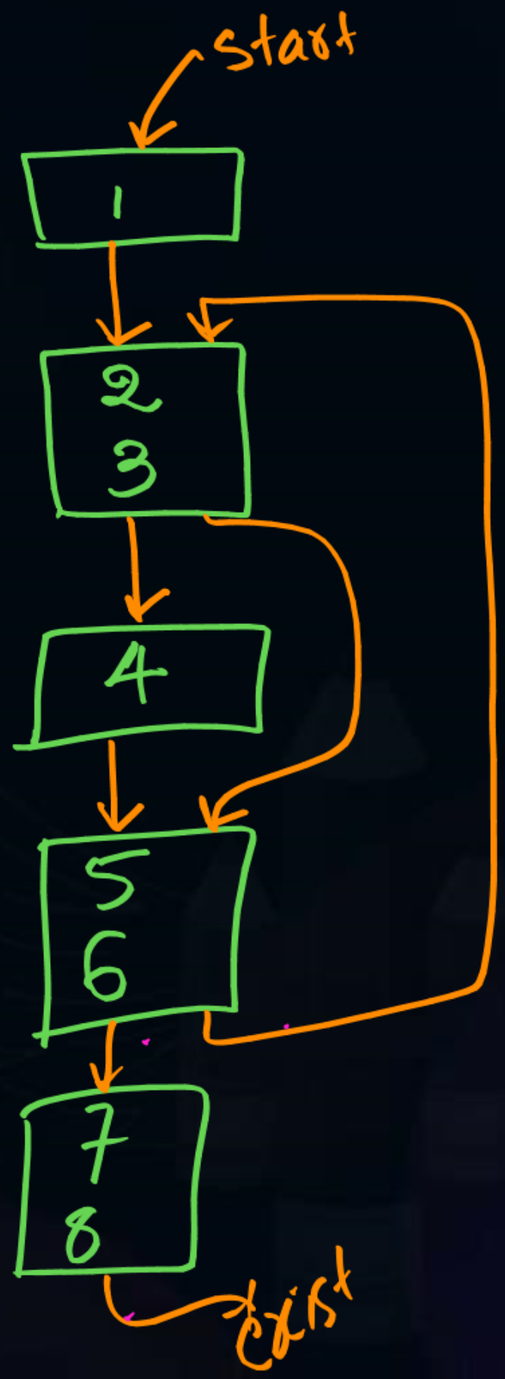
CCC : (+, b, d, c)

Address	Operator	Operand 1	Operand 2
XXX	+	a	b
YYY	*	XXX	c
ZZZ	+	YYY	d

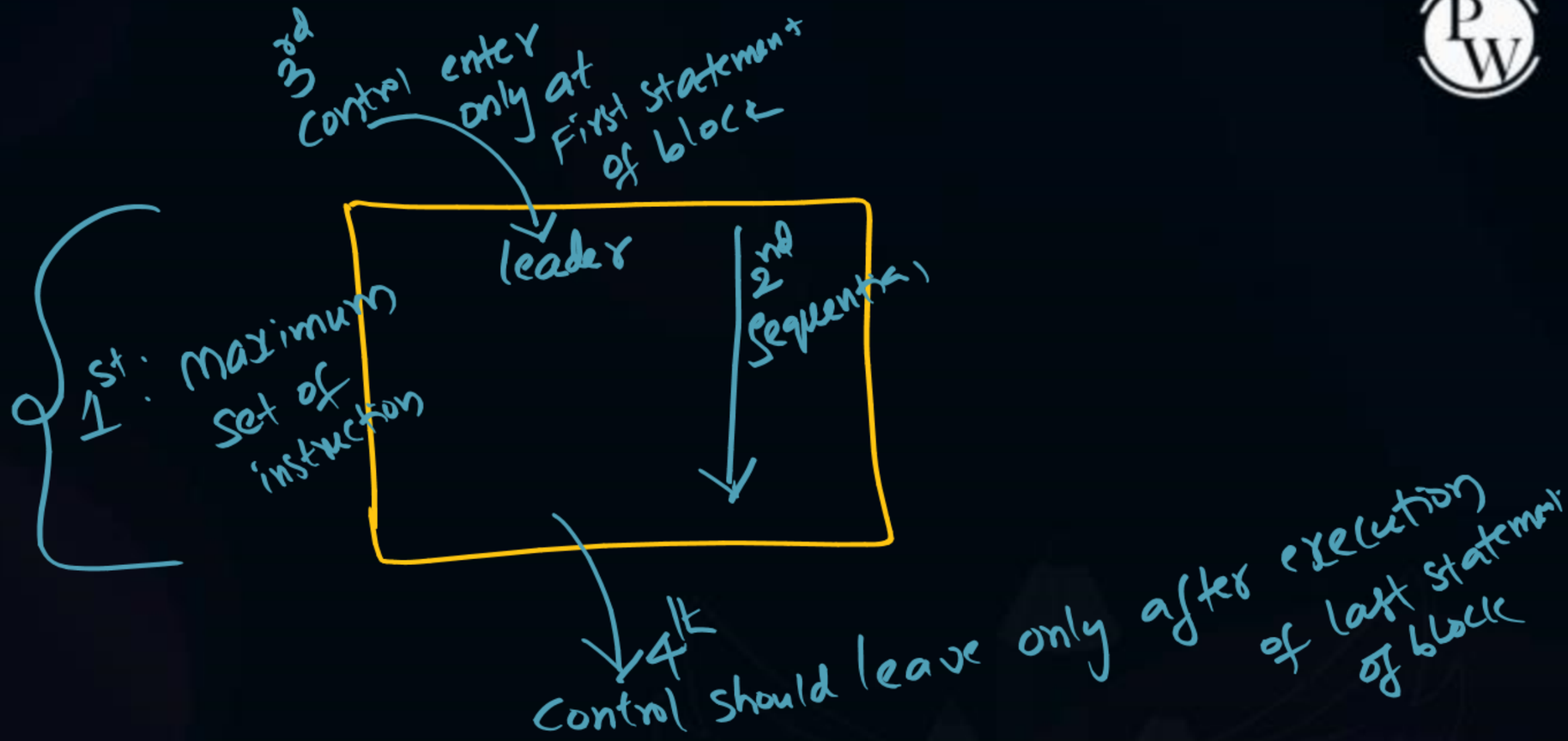
(7) Construct CFG for the following 3AC.

1. $x = a$
2. $y = x * c$
3. if $(y > 100)$ goto 5
4. $y = y - b$
5. $z = z + y$
6. if $(z < 500)$ goto 2
7. $a = x + y$
8. print (a)

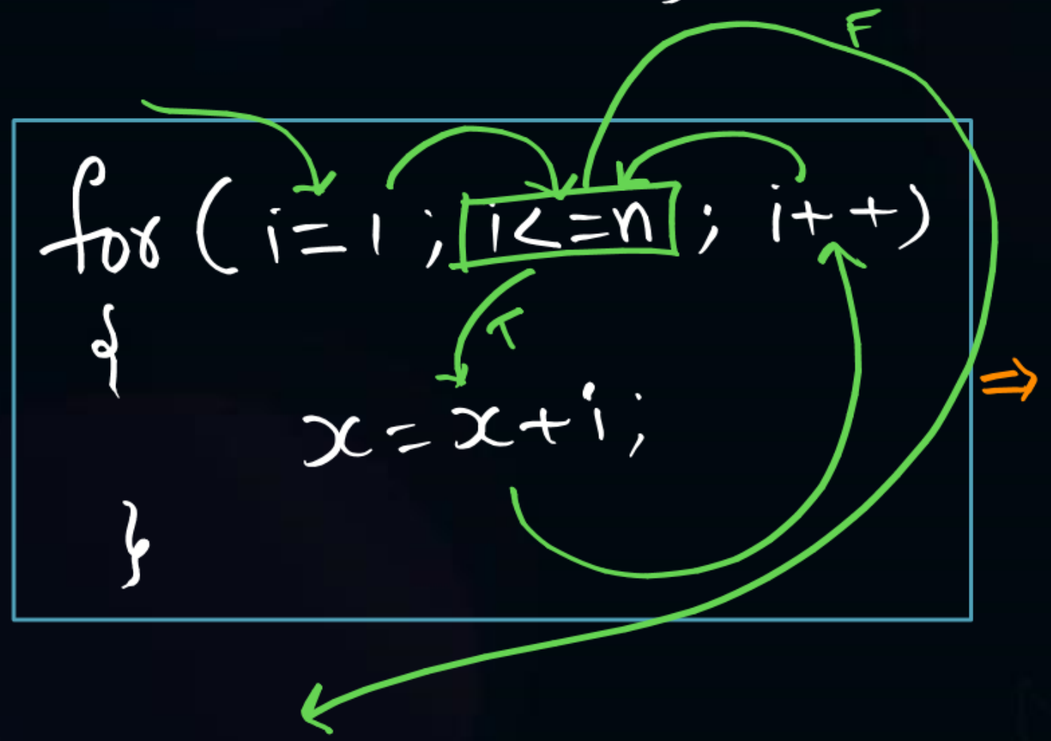




5 Basic Blocks
7 nodes
8 edges


3rd
Control enter only at First statement of block
leader
2nd Sequential
1st: Maximum Set of instruction
4th
Control should leave only after execution of last statement of block

(8) Convert the following loop into equivalent 3AC.



⇒

1) i=1
2) if(i>n) goto 5
3) x=x+i
4) i=i+1 , goto 2
5) exit.

(9) Match the following.

Group-I	Group-II
A. $x = 2 * 3$ (Constant Folding)	I. Copy propagation
B. $x = y + 0$ (Identity)	II. Constant Folding
C. $x = x * 2$ (Strength Reduction)	III. Strength Reduction
D. $x = 2;\ y = x + a$ (Copy propagation)	IV. Algebraic Law Identity

A ← II

B ← IV ($y+0$, y)

C ← III ($y*1$, y)

D ← I

(10)

```
x = a;
y = 10;
z = y + 2 - a;
if (A[i*2] > b)
    c = z + A[i*2]
```

Find possible code optimizations in above code.

Dead: ~~x = a~~, y = 10

z = 12 - a

$t_1 = i + i$

if A[t_1] > b

c = z + A[t_1]

1) Constant Folding
2) Copy propagation
3) Strength Reduction
4) Algebraic Laws [Identity, Cancellation]
5) Loop merge
6) Loop Code motion
7) Induction Variable elimination
8) Common Sub exp elimination.
9) Dead code elimination

Compiler Design

- Find TOKens
- First & Follow sets
  - LL(1) CFGs
  - LR CFGs
  - Ambiguous CFGs
- Evaluations of SDT
  - Types of Attributes
  - Definitions of SDT
- IRs
  - 3AC
  - CFG
- Code optimization
  - Constant Folding
  - Copy propogation
  - Strength Reduction
  - Common sub exp elimination
  - Algebraic simpli.
  - Dead code elim.
  - Loop optimization

THANK - YOU