K. J. Somaiya Institute of Engineering and Information Technology, Sion (E), Mumbai - 400 022.

Operator	Operand1	Operand2	Result
Uminus	C		t1
	t1		t2

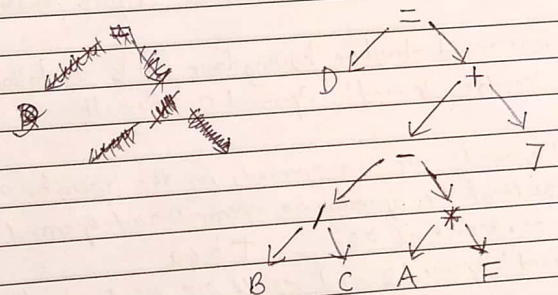
K. J. Somaiya Institute of Engineering and Information Technology, Sion (E), Mumbai - 400 022.

Q.1) Give the output of all phases of compiler for the following expression.
D = B/C - A*F + 7

Ans Lexical Analyzer

Identifier	Arithmetic Operator	Assignment Operator	Constant
D		=	7
B	/		
C	-		
A	*		
F	+		

~~Semantic Analyzer~~ Syntax Analyzer



Semantic analyzer
Here, we have to convert all ~~integer~~ numerical values and identifier into one type ie. float or integer to obtain the answer

Intermediate Code Generation

```
t1 = B/C
t2 = A*F
t3 = 7
t4 = t1-t2
t5 = t4+t3
C = t5
```

Department of Computer / Class- TE / Sem- VI / Sub- SPCC / Academic Year- 2018-19

Department of / Class- / Sem- / Sub- / Academic Year-

Academic Year- 2018-19

Q1) Give the output of ... following expression:
D = B/C - A*F + 7
Analyzer: Arithmetic operator Assignment operator Constant
=
7

Code Optimization

```
t1 = B/C
t2 = A*F
t3 = t1 + t2 + 7
C = t3
```

Code Generation

```
R1 ← B , R1 ← A , R1 ← R3 - R4 , C ← R1
R2 ← C , R2 ← F , R2 ← 7
R3 ← R1/R2 , R4 ← R1 * R2 , R1 ← R1 + 7
```

Q.2 Explain Quadruples, triples and indirect triples with examples?

Ans Quadruple

A quadruple is a record structure having four fields as follows:-

Operator Operand1 Operand2 Result

Where operand1, operand2 are two operands for the operator and result field contains the result of the operation on operand1 and operand2. For example, a := b+c will be represent as + b c a

The fields operand1, operand2 and result are pointers to the symbol table entries, therefore temporary names must be entered into the symbol table. The three-address statement x := a op b is represented by placing a in operand1, b in operand2, and x in result.

- Unary operators like arg1 := op arg2 or arg1 := arg2 does not use operand2.
- Procedure call like param use neither operand2 nor result.
- A conditional and unconditional jump puts the target label in result.
- Operand1, Operand2 and result are usually pointers to symbol entries.
- Quadruples may need to use many temporary names.

Example1:-

a := b * -c + b * -c

```
t1 := -c ,  t4 := b * t3
t2 := b * t1 ,  t5 := t2 * t4
t3 := -c ,  a := t5
```

	Operator	Operand1	Operand2	Result
(0)	Uminus	c		t1
(1)	*	b	t1	t2
(2)	Uminus	c		t4
(3)	*	b	t3	t5
(4)	+	t2	t4	a
(5)	:=	t5		

Advantage:- In quadruples it is easy to rearrange code for global optimization

Disadvantage: In this lots of temporaries are created, so lot of memory is wasted.

Triples

A triple is a record structure having three fields as follows:

Operator Operand1 Operand2

To avoid entering temporary names into the symbol table, we might refer to a temporary value by the position of the statement that computes it. If we do so the three address code can be represented with only three fields operator, operand1 and operand2 are either pointers to the symbol table or pointers into the triple structure. If refers to the symbol table for user-defined names or constants and to the triple structure for temporary values.

The triple A*B represents the infix notation ~~A*B~~ *AB.

Example a := b* - c + b* - c

$t_1 := c$
$t_2 := b * t_1$
$t_3 := c$
$t_4 := b * t_3$
$t_5 := t_2 + t_4$
$a := t_5$

	Op	arg1	arg2
(0)	Uminus	c	
(1)	*	b	(0)
(2)	Uminus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	Assign	a	(4)

Advantage:- In triples temporaries are implicit
Disadvantage:- In triples it is difficult to rearrange code.

Indirect Triples
Another implementation of three-address code that has been considered is that of listing pointers to triples, rather than listing the triples themselves. This implementation is naturally called indirect triples. In indirect triples we use an array statement to list pointers to triples in the desired order.

	Statements		#	Operator	Argument 1	
0	100	→	100	uminus	C	
1	101	→	101	*	B	100
2	102	→	102	uminus	C	
3	103	→	103	*	B	102
4	104	→	104	+	101	103
5	105	→	105	=	A	104

Advantages:-
- In indirect triples the temporaries are implicit and easier to rearrange code.
- A statement can be moved by reordering the statement list.

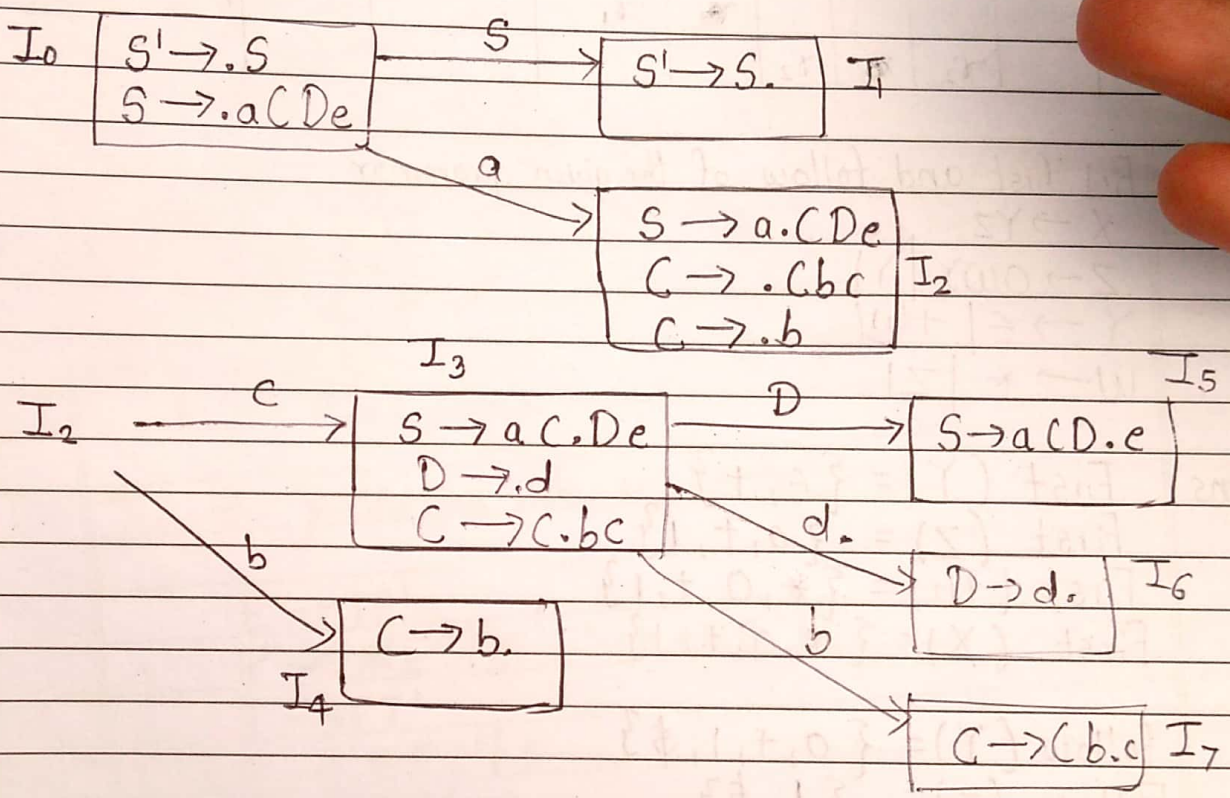
3) Construct LR(0) Set of items for the following grammar.
S → aCDe
C → Cbc | b
D → d

Ans Thus, production rules are,
1) S → a CDe
2) C → Cbc
3) C → b
4) D → d
~~Given that,~~ Let
S' → S
S → aCDe
C → Cbc
C → b
D → d

First (S) = {a}
First (C) = {b}
First (D) = {d}

Follow (S) = {$}
Follow (C) = {b, d}
Follow (D) = {e}

I0 Closure :- S' → .S



$I_5 \xrightarrow{e}$ S → aCDe. I_8

$I_7 \xrightarrow{c}$ C → Cbc. I_9

	Action						Goto		
	a	b	c	d	e	$	S	C	D
0	S2						1		
1						acc			
2		S4						3	
3		S7		S6					5
4		r_3		r_3					
5					S8				
6					r_4				
7			S9						
8						r_1			
9		r_2		r_2					

Q.4 Find first and follow of the given grammar

Ans
$X \rightarrow YZ$
$Z \rightarrow OWY \mid Y1$
$Y \rightarrow \epsilon \mid +W1$
$W \rightarrow * \mid Z1$

Ans
First (Y) = {ϵ, +}
First (Z) = {0, +, 1}
First (W) = {*, 0, +, 1}
First (X) = {*, 0, +, 1}

Follow (Y) = {0, +, 1, $}
Follow (Z) = {1, $}
Follow (X) = {$}
Follow (W) = {1, 0, +, $}

Q.5 Remove left recursion from the following grammar:-

Ans $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid id$

Left recursion = $A \rightarrow A\alpha \mid \beta$ then

$A \rightarrow \beta A'$

$A' \rightarrow \alpha A' \mid \epsilon$

1) $E \rightarrow E+T \mid T$

$A \rightarrow A\alpha \mid \beta$

Thus, $E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

2) $T \rightarrow T*F \mid F$

$A \rightarrow A\alpha \mid \beta$

Thus $T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

Thus, the production rule are as follows:-

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid id$