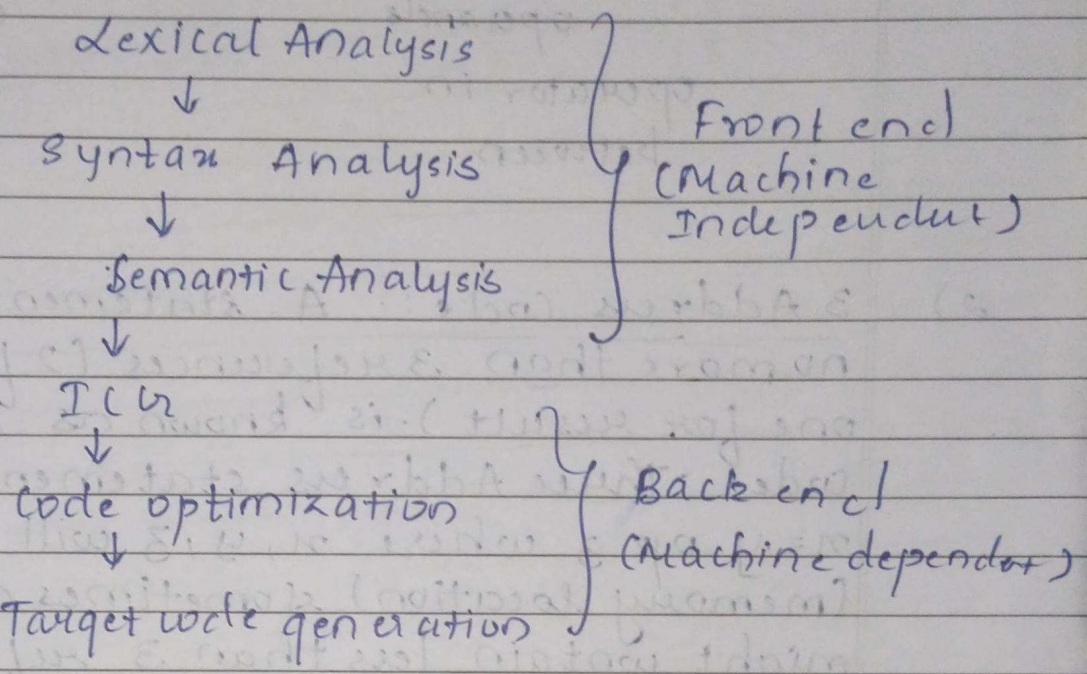
SPCC-08

DATE:

Aim:- Implement intermediate code generation using lex and yacc

Theory:-



If Machine code is generated directly from source code then for n target machine we will have optimizers and n code generator but if we'll have a machine independent intermediate code, we will have only one optimizer

Intermediate code, we will have only one optimizer. Intermediate code can be either language specific (Eg: byte code for Java) or language independent (3 Address code). The following are intermediate code representations

DATE:

1) Postfix Notation / Reverse Polish Notation / Suffix Notation

→ eg: infix = a+b (operands, operator in between) postfix = ab+ (operands) ← operator in end

2) 3 Address code: A statement involving no more than 3 references (2 for operands & one for result) is known as a 3 Address code. Three Address statement is of form $x = y$ op z where x, y, z will have address (memory location) sometimes a statement might contain less than 3 references but it's still called 3 Address Statement.

There are 3 ways to represent 3 Address code in.

- Quadruples
- Triple
- Indirect Triple

1) Syntax Tree: Condensed form of a passe tree. The operator & keyword nodes of the parse tree are moves to their parents and a chain of single production is replaced by single link in syntax tree. the internal nodes are operators and child nodes are operands.

DATE:

Triple:-

	op	arg1	arg2	Indirect triple
(0)	+	a	b	10
(1)	-	(0)	-	11
(2)	+	c	d	12
(3)	*	(1)	(2)	13
(4)	+	(0)	c	14
(5)	-	(3)	(4)	15