SPCC-08

Aim:- Implement intermediate code generation using lex and yacc

Theory:-

Lexical Analysis
↓
Syntax Analysis
↓
Semantic Analysis

} Front end (Machine Independent)

↓
ICG
↓
Code optimization
↓
Target code generation

} Back end (Machine dependent)

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: bytecode for Java) or Language independent (3 Address code). The following are intermediate code represent's

1) Postfix Notation/ Reverse Polish Notation/ Suffix Notation

→ eg: infix = $\underbrace{a+b}_{\text{operands}}$ operator in between

postfix = $\underbrace{ab}_{\text{operands}}$+ ← operator in end

2) 3 Address code:. A statement involving nomore than 3 references (2 for operands & one for result). is known as a 3 Address code. Three Address statement is of from x = y op z where x, y, z will have address (memory location) sometimes a statement might contain less than 3 references but its 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 parse 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.

Triple:-

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