Compiler Design (CSE 306)

Tutorial - 1

Set 2

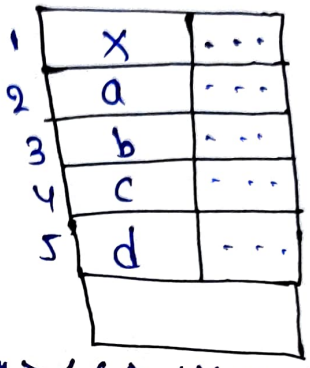
V. Sai Harshith
AP20110010045
CSE-A

1. Write a output at all phases of the compiler of the following code snippet:

$$X = (a+b)^* (c+d)$$

Sol)

$$x = (a+b)^*(c+d)$$



↓

Lexical Analyzer

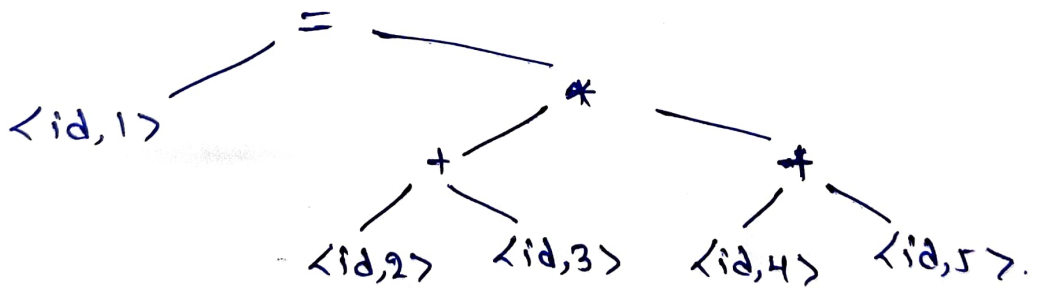
↓

<id,1><=><(><id,2><+><id,3><)><*><(><id,4><+><id,5><)>

↓

Syntax Analyzer

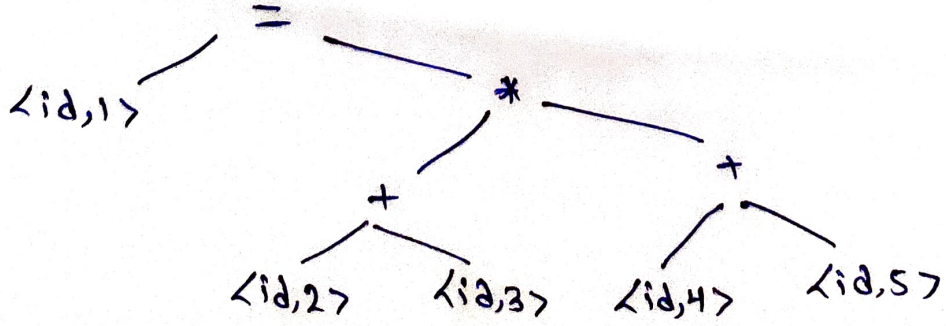
↓



↓

Semantic analyzer

↓



↓

Intermediate code Generator

↓

~~t1 = id2 + id3~~
~~t2 = id4 + id5~~
~~t3 = t1 * t2~~
~~id1 = t3~~

t1 = id2
t2 = id3
t3 = t1 + t2
t4 = id4
t5 = id5
t6 = t4 + t5
t7 = t3 * t6
id1 = t7.

↓

code optimizer

↓

t1 = id2 + id3
t2 = id4 + id5
id1 = t1 * t2

↓

Code generation

converts the code to machine level / assembly level.

↓

ADD R1, id1, id2
ADD R2, id3, id4
MUL R3, R1, R2

2. Define Lexeme, token and Pattern. Identify lexemes that make up the tokens in the following Program segment and indicate the corresponding Pattern.

```
int sum(int i, int j)
{
    int temp;
    temp = i+j;
    return temp;
}
```

lexeme:-

It is a sequence of characters from the input source Program and that matches a Pattern. It is classified by a token.

Tokens:-

Token is a sequence of characters that can be treated as a single logical entity.

Patterns:- It specifies a set of rules that a scanner follows to create token.

int, return } → keywords → sequence of chars which has a specific function.

sum, i, j, temp } → identifiers → starting with a character, ending.

{, }, ; → delimiters → character that marks the beginning or end of a unit data.

+, = → operators.

lexeme	token	Pattern
int	Keyword	sequence of chars
sum	Identifier	sequence of chars & nums
(	delimiter	open paranthesis
int	Keyword	sequence of chars
i	identifier	sequence of chars & nums
int	~~delimiter~~ Keyword	sequence of chars
j	identifier	sequence of chars & nums
)	delimiter	close paranthesis
{	delimiter	open braces
int	Keyword	sequence of chars
temp	identifier	sequence of chars & nums
;	delimiter	semicolon
temp	identifier	sequence of chars & nums
=	Assignment operator	=
i	identifier	sequence of chars & nums
+	operator	arithmetic operators
j	identifier	sequence of chars & nums
;	delimiter	semicolon.
return	Keyword	sequence of chars
temp	identifier	sequence of chars & nums
;	delimiter	semicolon.
}	delimiter	close braces.

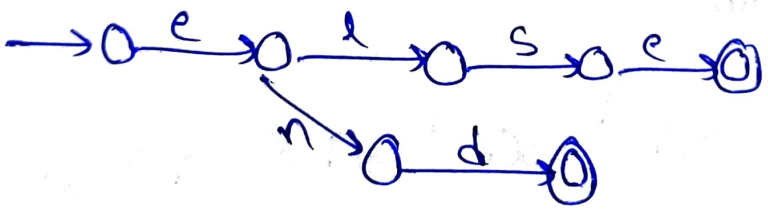
3. b write a regular expressions and transition diagrams for the following.

a. keywords : else, end

regular expression:- else :- else stmt;

end :- fun (stmt, end = "character")

Transition diagram:-



b. Assignment operators:- (=, +=, -=, *=, /=, %=)

Regular expression:-

= → <letter><=><integer> | <letter><=><=><letter>

+= → <letter><+><=><integer>

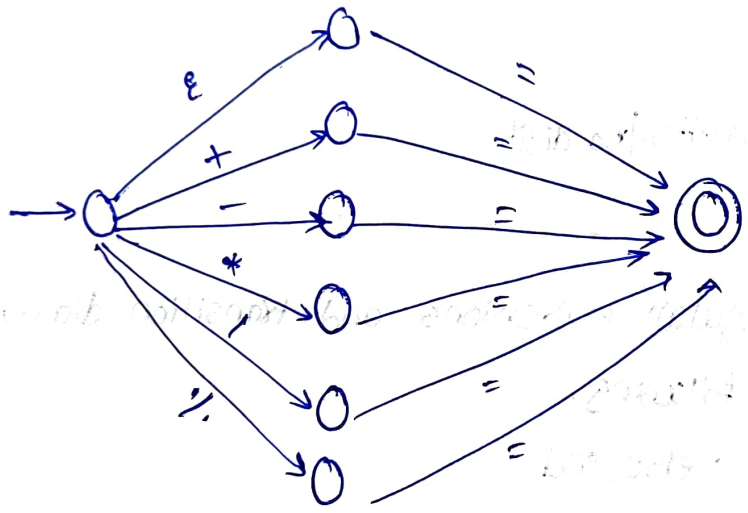
-= → <letter><-><=><integer>

= → <letter><><=><integer>

/= → <letter></><=><integer>

%= → <letter><%><=><integer>

Transition diagram:-

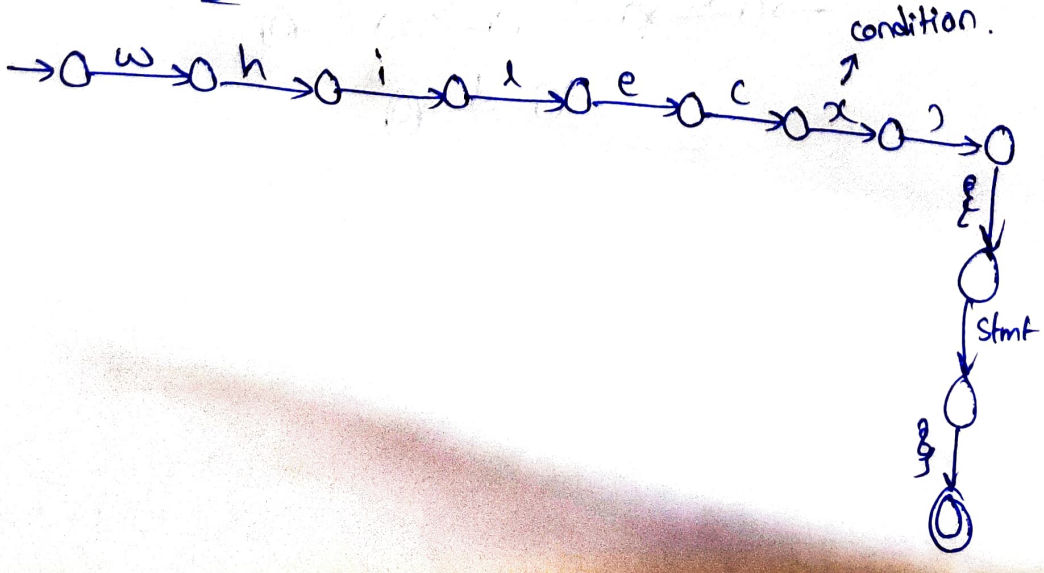


c) while statement:-

Regular expression:- while (stmt) { stmt };

Stmt → conditional statement / expression

Transition diagram:-



4. Write LEX Program to add line number before each line in a file.

```
%{
#include<
#include<
int ln=0;
%}
%%
"\n"{ }
.*{ln++; fPrintf(yyout,"\n%d: %S",ln,yytext);}
%%
main()
{
yyin = fopen("try1.txt","r");
yyout = fopen("try2.txt","w");
yylex();
return 0;
}
int yywrap()
{
}
```

Output:-

try1.txt

```
Hello world
lex Program to add line number
before each line
in a file.
```

try2.txt

```
1: Hello world
2: lex Program to add line
number
3: before each line
4: in a file.
```