Unit 1

PART-A

1) benifits of seperating lexical Analysis from syntax Analysis:-

1) simple design

- seperation allows the simplification of one or the other

ex:- A parser with comments or white spaces is more complex

2) compiler efficiency is Improved

- optimization of lexical Analysis because large Amount of time is spent on reading the source program and partitioning into token.

2)

Lexeme	Token	Patterns
int	keyword	
diff	identifier	
,	Symbol	
int1	Identifier	
int2	identifier	
;	SP Symbol/semicolon	
if	keyword	
>	operator	
=	operator	
-	operator	
else	keyword	

Scanned with CamScanner

3) The error recovery for the given program is:-
* Transforming two Adjocent characters
ex:- esle → else

4) 1) Structure editor
2) Pretty printers
3) Static checkers
4) Interpretors.

5) * It is non-trival test of language
* This Technique makes the compiler self-hosting compiler
* compiler development can be done in higher language is also compiled.
* compiler developer only needs to know that language in which target code is compiled.

6) Cousings of compiler
1) preprocessor
2) Assembler
3) Loader and Link editor.

7) The two parts of compilation is
1) Analysis
2) synthesis
1) Analysis:- this include lexical and Syntatic Analysis and creation of symbol table, sematic Analysis and Intermidiate Code generation. and Includes error handling on all these phases

Scanned with CamScanner

(DEEMED TO BE UNIVERSITY)

7) Synthesis :- It include code optimization and code generation along with necessary error handling operations.

8) structure editor :-
* Takes as input a sequence of commands to buid a source program
* It not only performs the text creation and modification functions of an ordinary text editor, but it also analyze the program text, putting an appropriate hierachical structure of the source program.
ex :- It can supply keywords Automatically :-
while... do and begin... end.

9) i) lexical
ii) syntax
3) Semantic
4) Intermediate code
5) code optimization
6) Code generator.

10) Pretty printers
* A pretty printer Analyze a program and prints it in such a way that the structure of the program becomes clearly visible.
ex :- comments may Appear in special fonts

Static checkers :-
* A static checker reads a program, Analyze it and attempts

Scanned with CamScanner

to discover potential bugs without running the program.
ex:- A static checker may detect the parts of the source program can never be executed.

11) $L = \{w \in \{a,b\}/w \text{ in } abb\}$
$= a,b \text{ (or) } a|b$
$= (a+b)^*abb$
$= \{abb, aabb, aabbbb\}$

12) Sentinal:- The sentinal is a special character that cannot be part of the source program, and natural choice is the character is eof.

Usage
eof retains its use as a marker for the end of the entire input. Any eof that appears other than at the end of buffer means that the input is at an end.

13) Rational preprocessor:-
These processors change older languages with more modern flow-of-control and data structre facilities.

14) 1) macre processing
2) file Inclusion
3) Rational pre processors
4) language extension

Scanned with CamScanner

15) 1) Deleting an extranious character
2) Inserting a missing character
3) Replacing Incorrect character with correct character
4) Transforming two adjecent characters
5) panic mode recovery:- Deletion of sucessive char from the token until error is resolved

16) one buffer
two-buffer scheme.

PART-B & C

17) cousins of compiler
1) Preprocessor
2) Assembler
3) Linker and Loader

1) Preprocessor:- A preprocessor is a program that process the input data to produce output that is used as Input to another program
* The output is said to be preprocessed form of the input data, which is often used by some subsequent programs Like compilers.

They may perform some function like:-
* macro procession
* file Inclusion
* Rational preprocessors
* Language extension.

Scanned with CamScanner

1) macro processing :- A macro is a pattern which certain sequence of input is mapped to output sequence according to defined procedure.

2) file Inclusion :- preprocessor includes header files into program text files. when preprocessor finds #include directive it replaces it by the entire content of the specified file.

3) Rational preprocessors :- These preprocessor change older languages with modern flow-of-control and data structring facilities.

4) Language extension :- This processors attempt to add capabilities to the language by what amounts to built-in macros.

2) Assembler :-

Assembler creates object code by translating assembly instaruction mnemonics into machine code.

Two types of Assemblers :-

* one-pass assembler go through the source code once and assume that all symbols will be defined before any instruction references them.
* Two-pass assembler creates a table with all symbols and their values in first pass and then use table in second pass to generate code.

3) Linker & Loader

A linker or Link editor is a program that take one or more objects generated by compiler and combines them into a single executable program.

Three tasks of Linker

1. Searches the program to find library routine used by a program.
2. Determine the memory locations and realocate the instructions
3. Resolves refrences among files.

Scanned with CamScanner

A loader is part of OS that is responsible for loading programs in memory.

18) Compiler construction Tools

1) Parser Generators
* These produce syntax Analyzers, normally from output that is based on CFG.
* It consumers a large fraction of the running time of a compiler.
* ex:- YACC.

2) Scanner Generators:-
* These generate lexical Analyzers, normally from a specification based on RE
* The basic organization of lexical Analyzer is based on finite Automata.

3) Syntax-Directed Translation
→ These produces routines that walk the parse Tree and generate Intermediate code as result
→ Each Translation is defined in terms of Translation as its neighbour nodes in the Tree.

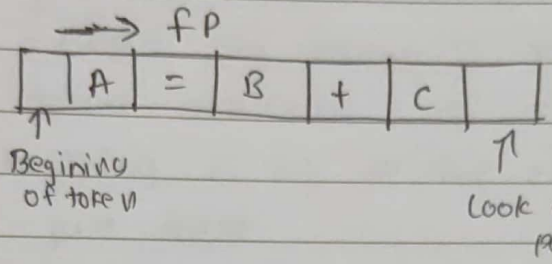
4) Automatic code generators:-
→ It takes a collection of rules which translate Intermidiate code to machine code.

5) Data flow engine:-
→ It does code optimization using data-flow Analysis, that is the gathering of information about how values are transmitted from one part of program to each other part.

Scanned with CamScanner

19) Input Buffering

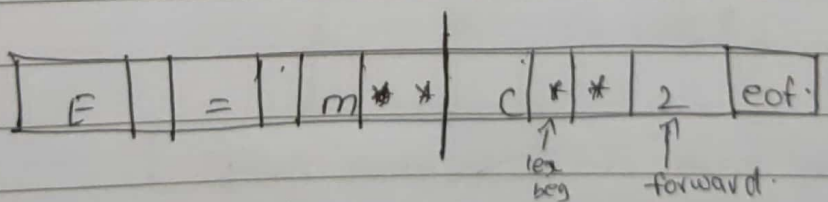
we often have to looke one or more characters beyond the next lexeme before we can be sure we have right lexem. As we know that As characters are read from left to right, each character is stored in the buffer to form meaning full token



we have two input buffering techiques:-
one-buffer scheme, two-buffer scheme

* we introduce two buffer scheme that handles large look Ahead safely. we then consider an Improument involving "sentinals" which is used to check end of buffer.

ex:-

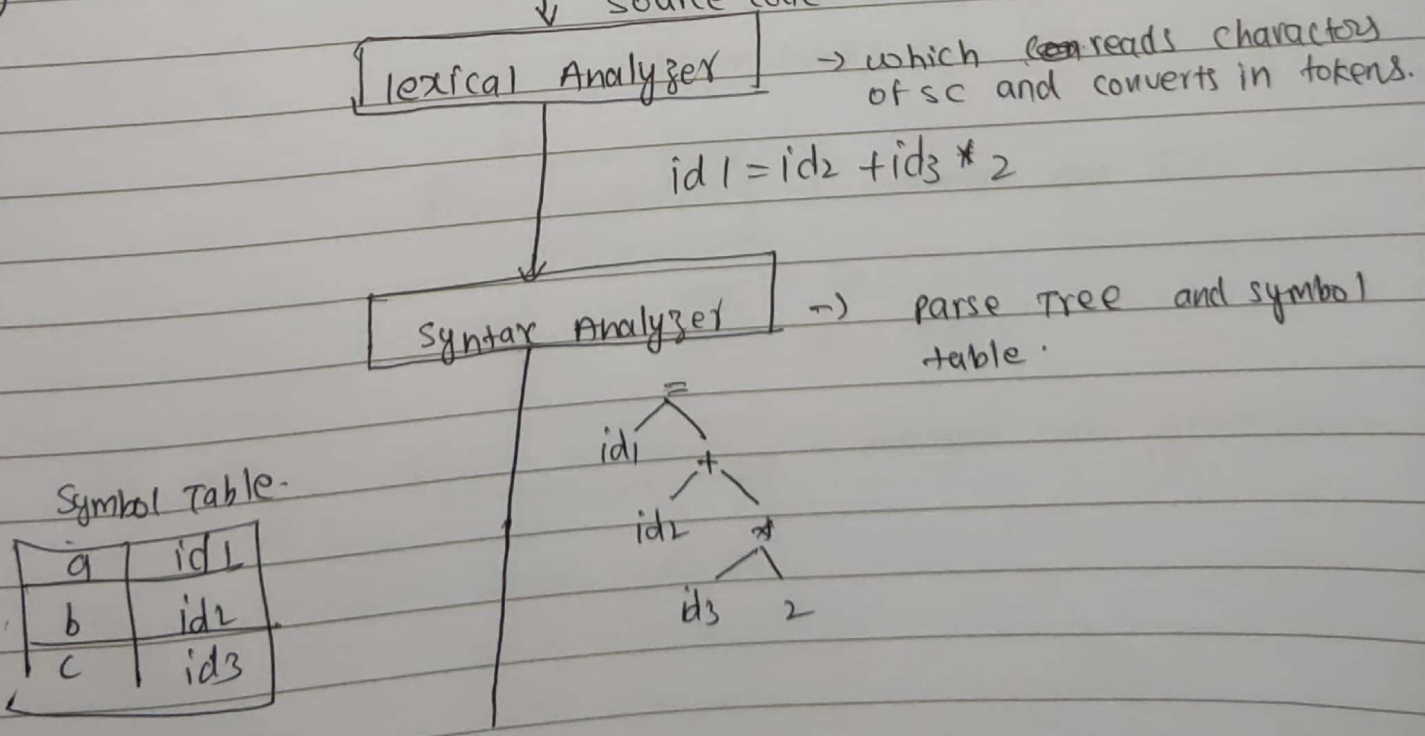


- A buffer is divided int two N-character haves.
* Each buffer is of the same size of N
* using one system read command we can read N char into a buffer
* if fewer than N char remain in input file, then special character, represented by eof, marks the end of source file.

Scanned with CamScanner

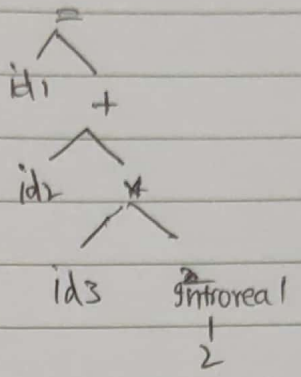
* two pointers to the input is maintained.
  1) lexeme begining :- which marks the beginnig of current lexeme
  2) forward :- scands ahead until a pattern match is found. one the next lexeme is determined forward is set to the charactor at its right end.
* The string of characters b/w the two pointers is current lexeme. After the lexeme is recordes as an Attribute value of a token returned to the parser.
  lexeme begining is set to the character immediatly after the lexeme is just found.
* eof is special charactor which tells the end of buffer or end of input from source file.

20) example $a = b + c * 2$



Scanned with CamScanner

Semantic analyzer → Type checking is done



Intermediate code generator → IC generation with SDT

```
temp1 = inttoreal(2)
temp2 = id3 * temp1
temp3 = id2 + temp2
id1   = temp3
```

Code optimizer → optimize the IC

```
temp1 = id3 * 2.0
id1 = id2 + temp1
```

Code generator → target code generation.

```
MOVF  id3, R2
MULF  #2.0, R2
MOVF  id2, R1
ADDF  R2, R1
MOVF  R1, id1.
```

Scanned with CamScanner