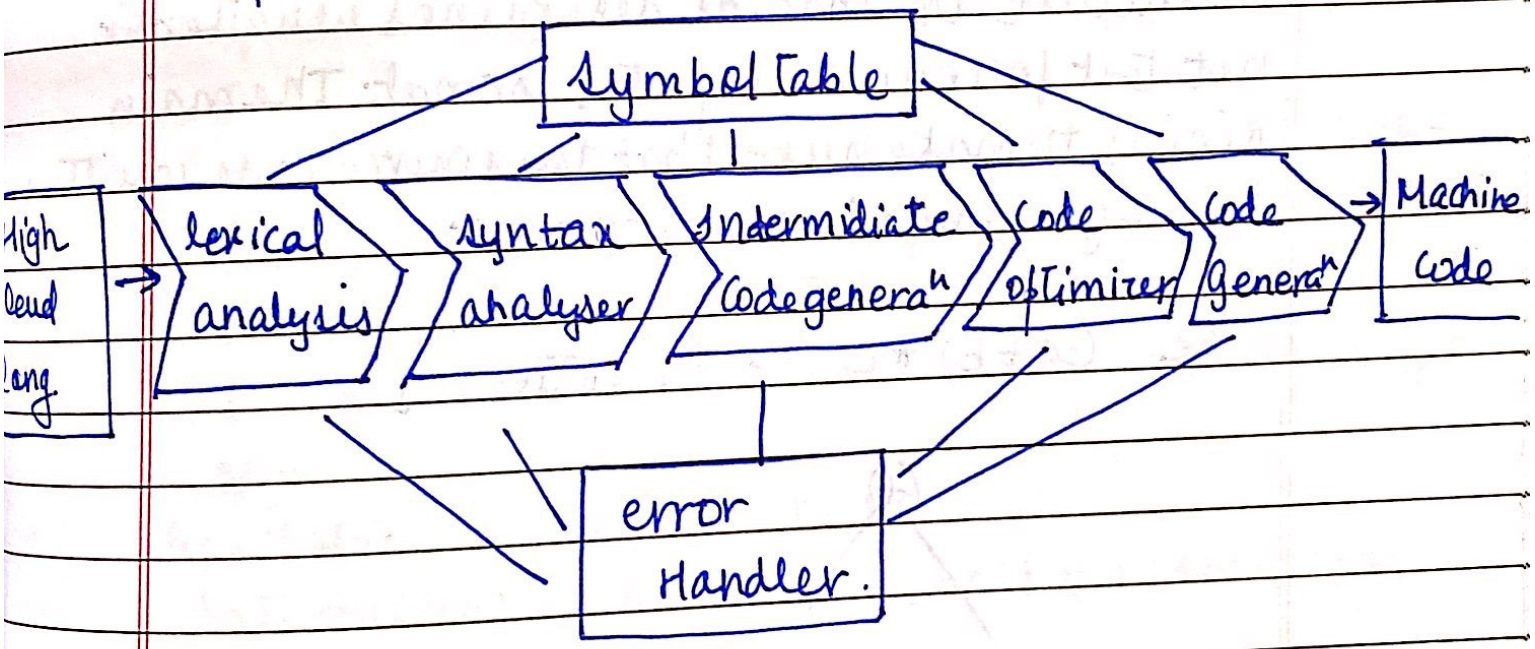
Aishwarya Kulkarni
PE24 DIV-E
Batch E2

Software system Compiler - Theory

Q.1 Explain the phases of compiler with suitable example.



PHASES OF COMPILER.

a) Lexical Analyser : It is the first phase when compiler scans the source code. This process be left to right by character & group these characters into tokens.

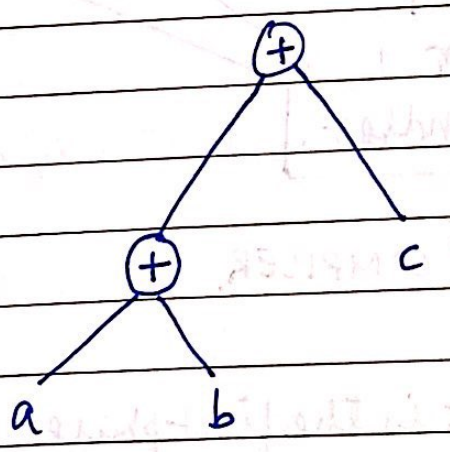
Eg. $X = y + 10$

Tokens:

X → identifier Y → identifier
'=' → Assignment operator
'+' → Addition operator
10 → Number

b) Syntax Analysis - It is all about discovering structure in code. It determines whether or not text follows an expected format. The main aim is to make sure that the source code is written by programmer is correct or not

Eg. (a+b) * c → Parse Tree



c) Semantic analysis: Checks the semantic consistency of the code. It uses the syntax tree of the previous phase along with

the symbol table to verify that the given source code is semantically consistent. It also checks whether the code is conveying an appropriate meaning.

Eg. float x = 20.2;
float y = x + 30;

- In this the semantic analyser will type cast int 30 to float 30 to float 30.0 before multiplication.

d) Intermidiate code generation: It is b/w the high-level & machine level language. This intermidiate code needs to be generated in such a manner that makes it easy to translate it into the target M.C

Eg. total = count + rate * 5.

Intermidiate code with the help of address code method is:

t1 = int_to_float(5)
t2 = rate * t1
t3 = count + t2
total = t3.

e) Code optimization : This phase removes unnecessary code line & arranges the sequence of statement to speed up the execution of program w/o wasting resources.

Eg. a = inTo float (10)
b = c * a
d = e + b
f = d.

→ can become : b = c * 10.0
f = e + b.

f) Code generation : It gets input from the code optimization phase & produces the page code or object code as a result. The objective of this phase is to allocate storage & generate relocatable machine code.

Eg. a = b + 60.0.

· would possibly be translated to registers:

MOVF a, R1
MULF #60.0, R2
ADDF R1, R2.

Q.2 Define & give examples: Tokens, lexemes, Patterns:

- **Token** : It is a sequence of characters that can be treated as a single logical entity. Typical tokens are,s :
(1) Identifier (2) keywords (3) operators (4) special symbols (5) Constants.

- **Pattern** : A set of strings in the input for which the same token is produced as output. This is called as pattern associated with token.

- **Lexeme** - A lexeme is a sequence of characters in the source program that is matched by the pattern for a token.

Examples -

Token	Lexeme	Pattern
const.	const	const
if	if	if
relation	<, <=, <>, =, >	< or <= or = or <> or =>
i	pi	any char. b/w "and" & "except"
num	3.14	
literal	"core"	Pattern

Q3. What is the purpose of symbol table?

Ans. Symbol table is an important data structure created & maintain by compilers in order store into about the occurance of various by entities such as variable names, function names, objects, interfaces, etc.

Q.4. Write a LEX program to count the number of vowels & constants in given string.

```
%{
    int vow_count = 0;
    int const_count = 0;
%}

%%
[aeiouAEIOU] { vow_count++; }
[a-zA-Z] { const_count++; }
%%
int yywrap() {}
int main()
{
  printf("Enter the string of vowels &
          consonants:");
  yylex();
```

```
printf("NO. of vowels are : %d\n", vow_count);
printf("NO. of consonents are : %d\n", const
                                    -count);
return 0;
}
```

Output:

Enter the string of vowels & consonents: Hello World
NO. of vowels : 3
NO. of consonents : 7.