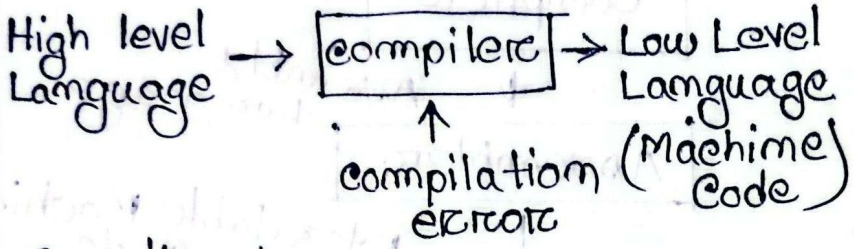
TOPIC NAME : DAY : TIME : DATE : / /

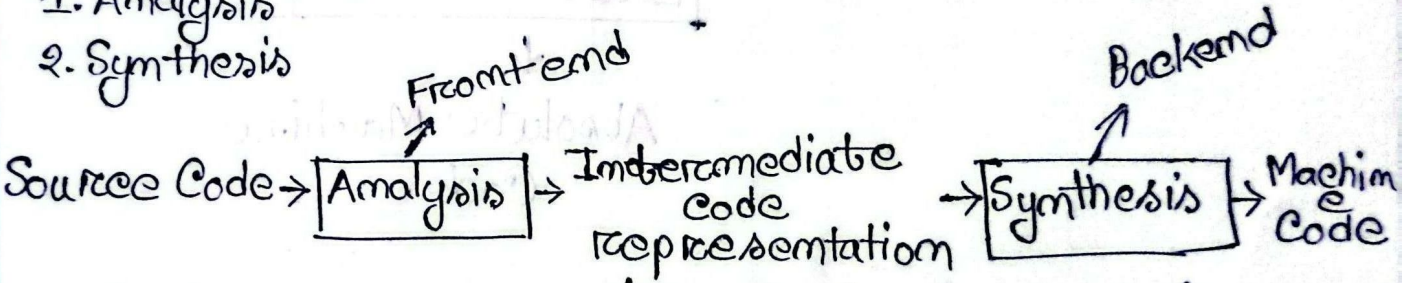
Introduction To Compilers

⇒ The compilers is a software that converts a program written in high level language to a low level language.



Compilers has 2 major phase:

1. Analysis
2. Synthesis



Analysis phase parts:

1. Lexical Analyzers
2. Syntax Analyzers
3. Semantic Analyzers
4. Intermidiate Code Generation.

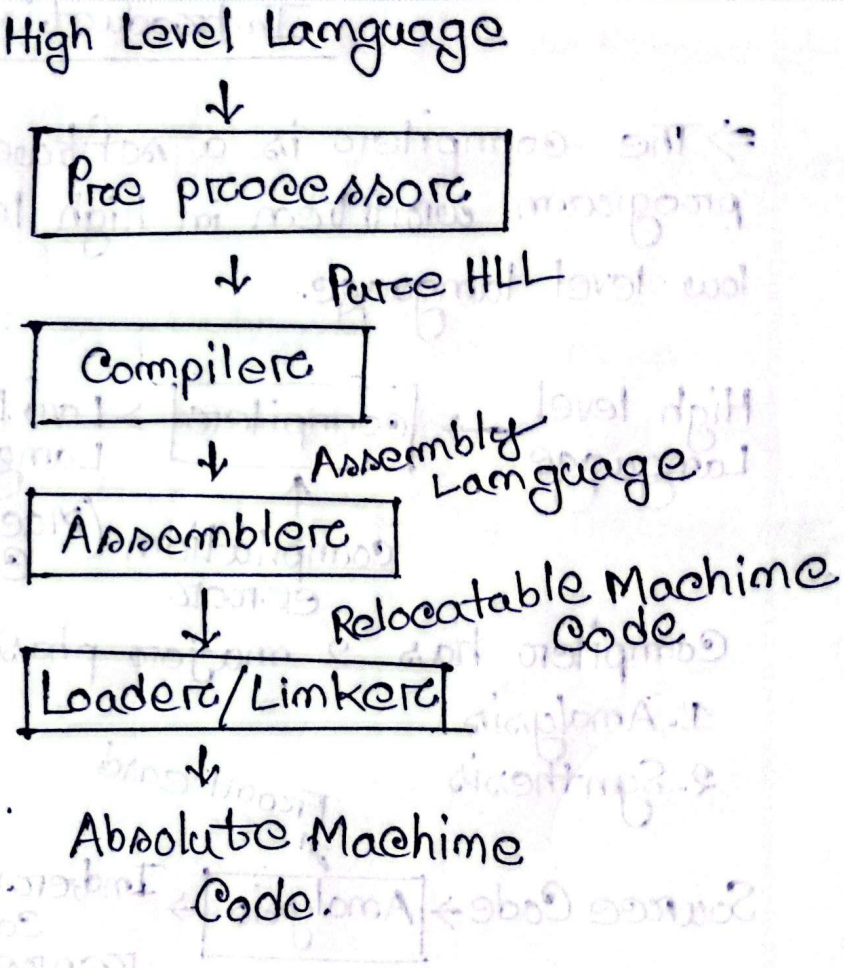
Synthesis phase parts:

1. Code optimizers
2. Target code generation.

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

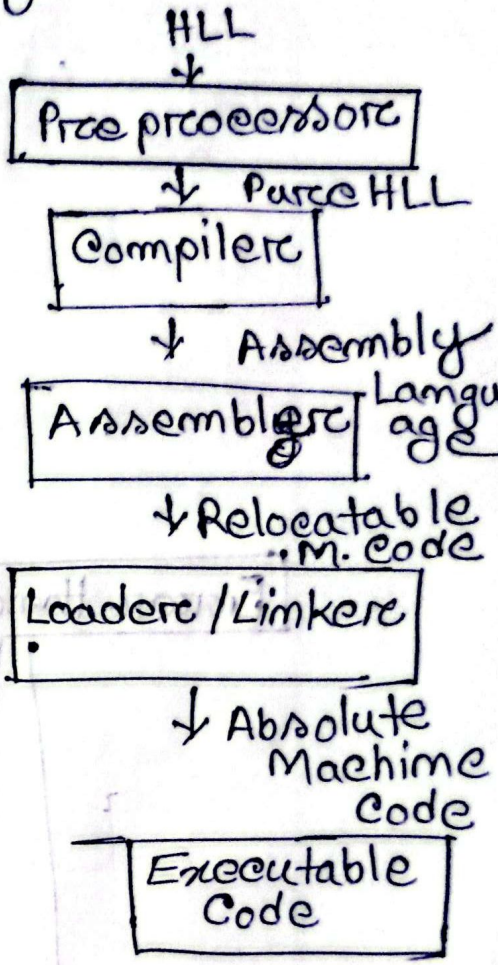


GOOD LUCK

TOPIC NAME : DAY :
TIME : DATE : / /

Language Processing System

```
#include <stdio.h>
#define Pi 3.1416
int main(){
    int radius=21;
    int area;
    area = Pi * radius * radius;
    printf("%d", area);
    return 0;
}
```



Linker: is a computer program that combines and links many object files to create one executable file.

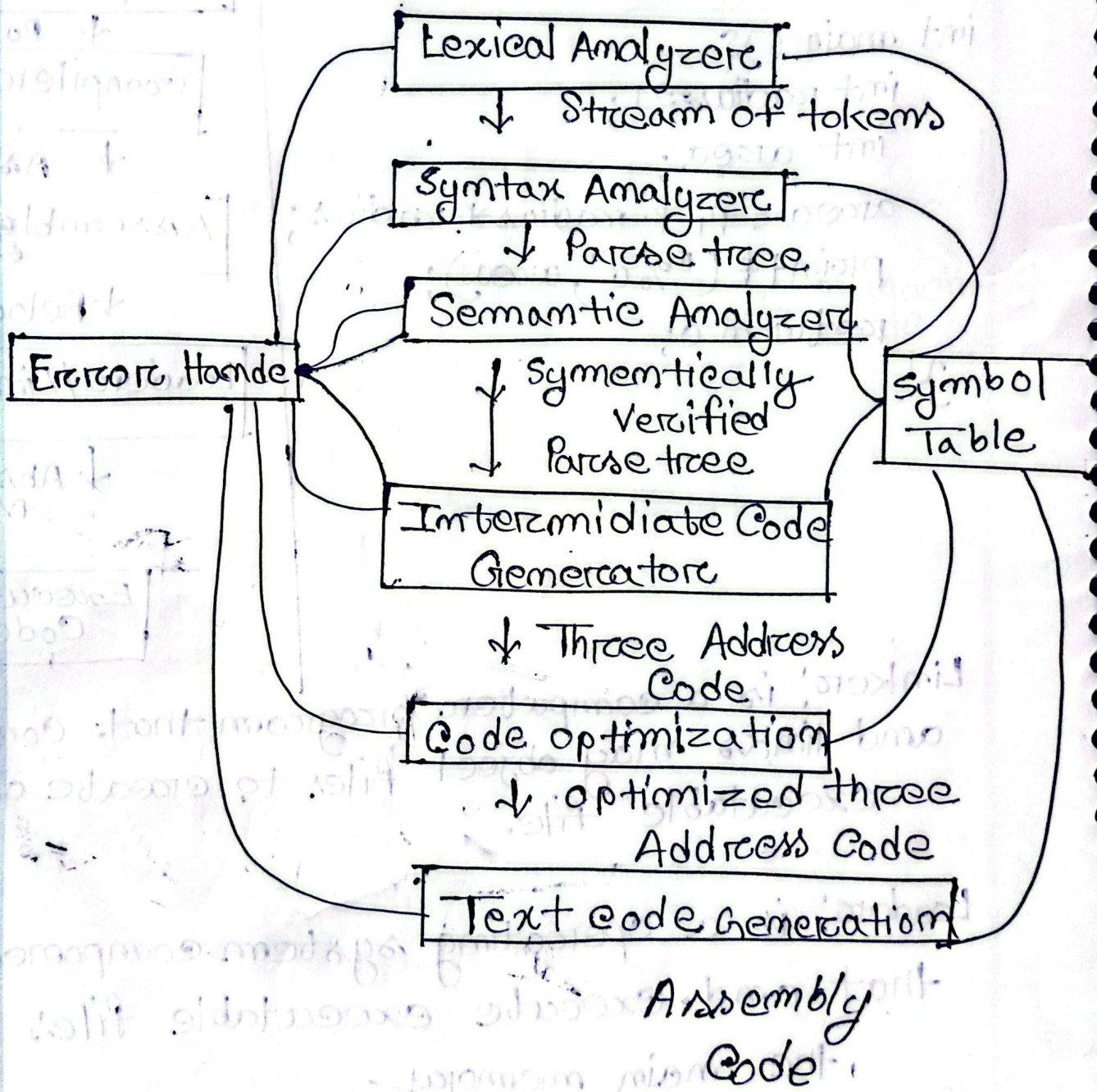
Loader: is a operating system component that and execute executable files into the main memory

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Phase of Compilers



"GOOD LUCK"

TOPIC NAME : _____ DAY : _____ TIME : DATE : / /

① int x, a, b, c

x = a + b * c

Symbol table

S.No	Variable	Type
1	x	int
2	a	int
3	b	int
4	c	int

2. Lexical Analyzer

<id,1> = <id,2> + <id,3> * <id,4>

id = identifier

x : indentifier

= : Assingment operator

a : identifier

+ : Addition operator

b : identifier

* : Multiplication operator

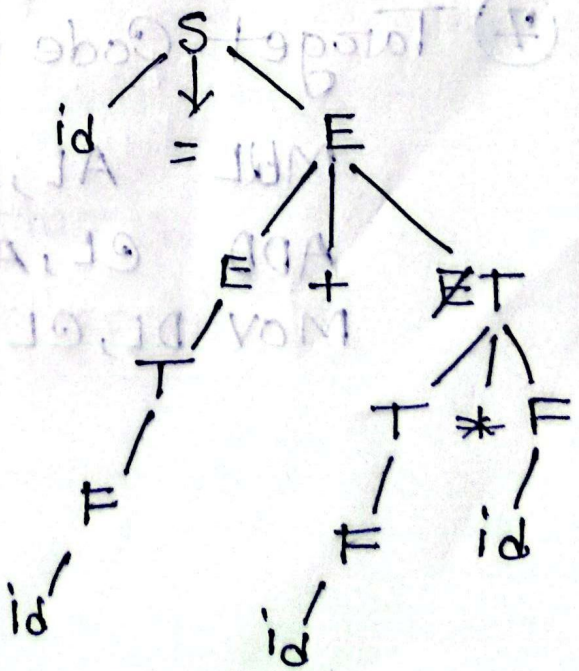
c : identifier

③ S → id = E

E → E + T | T

T → T * F | F

F → id



GOOD LUCK

TOPIC NAME : DAY : TIME : DATE : / /

④ Symantic Analyzerc

Symentically verified Parse tree

⑤ Intermidiate Code Generation

$t_1 = b * c$

$t_2 = t_1 + a$

$x = t_2$

⑥ Code Optimization

$t_1 = b * c$

$x = a + t_1$

⑦ Target Code Generation

```
MUL  AL, BL
ADD  CL, AL
MOV  DL, CL
```

GOOD LUCK

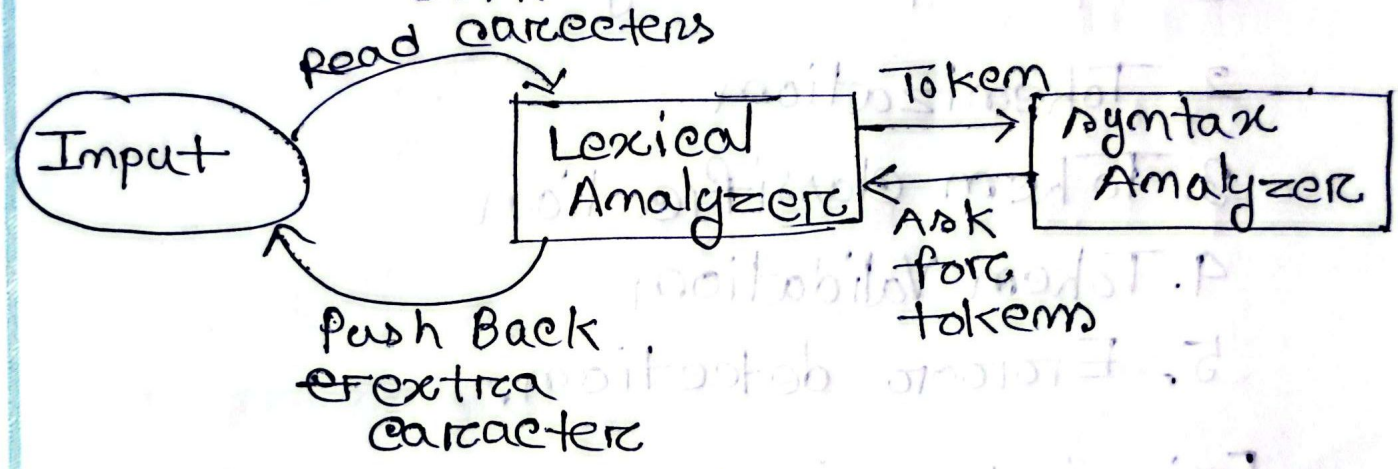
TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Lexical Analyzer

It is the first phase of compiler also known as scanner. It converts pure HLL into sequence of tokens.

→ Lexical Analyzer can be implemented with the DFA.



Token: is a sequence of carecters that can be thread as a unit in grammers of programming language

- keywords
- Identifiers
- operators
- constants
- Punctuation.

GOOD LUCK

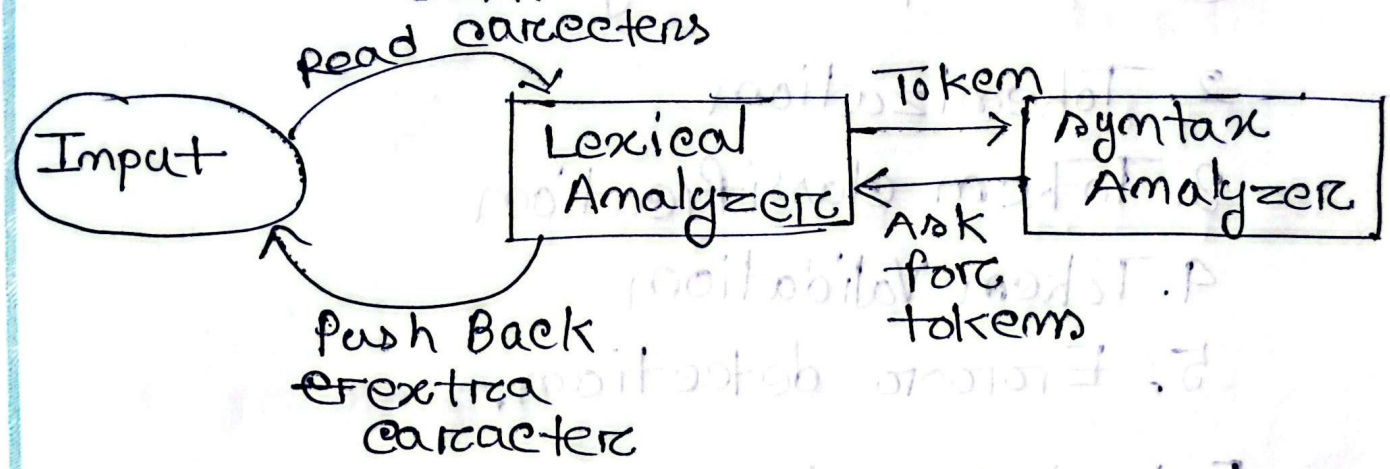
TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Lexical Analyzer

It is the first phase of compiler also known as scanner. It converts pure HLL into sequence of tokens.

→Lexical Analyzer can be implemented with the DFA.



Token: is a sequence of carcecters that can be thread as a unit in grammers of programming language

- keywords
- constants
- Identifiers
- Punctuation.
- operators

GOOD LUCK

TOPIC NAME : DAY :
TIME : DATE : / /

int a = 10 ;
- int → keyword
- a → identifier
- = → operator
- 10 → constant
- ; → punctuation

How lexical Analyzer works!

(1) input processing

2. Tokenization
3. Token classification
4. Token Validation
5. Error detection.

Ex:
```
int main ()
{
    int a, b;
    a = 10;
    return 0;
}
```
Token = 18

GOOD LUCK

TOPIC NAME : DAY : TIME : DATE : / /

Count Number of Tokens

```
int main ()  →4
{
 int A-= 10 , B=20, sum;  →16
 sum = A+B;  →22
 printf ("%d", sum);  →29
 return 0;  →32
} -33

int primenumber (int number) →6
{
for (i==2; i<=number/2; i++) →25
{ if (number % i!=0) →34
   continue →35
 else
   return 1;  →39
 }
 return 0;
} →44
```

GOOD LUCK

TOPIC NAME : DAY :
TIME : DATE : / /

```
int main ()→48
{
int num=7, res=0; →58
res = primenumber (num); -65
if (res == 0) →71
  printf ("%d", num); →78
 else→79
printf("%d not prime", num);
} →87
```

TOPIC NAME : DAY :
TIME : DATE : / /

1. int main) (→ 4
2. printf ("the value is %d", id) ; → 7
3. int a == 20 ; → 5
4. int b<=10 ; → 5
5. i n+ a=5 ; → 6
6. char a ='f' ; → 6
7. char * a = "xyz" ; → 6
8. int /* Alamin */ d = 100 ; → 5
9. in /* Alamin */ + d = 100 ; → 5
10. int /* abc */ * xyz ; → 4
11. A = = = B ; → 5
12. A = B + + + + + c ; → 8
13. ~~A = B = = =~~ A = B = = = c ;
14. A = B + = + + = = = c ; → 10

GOOD LUCK