📘 Compiler Design – Unit I Notes (RGPV Exam Oriented)

🎯 2 Days Before Exam Revision Notes 🔥

🌟 UNIT–I Syllabus

- Introduction of Compiler
- Major Data Structures in Compiler
- Bootstrapping & Porting
- Compiler Structure
- Analysis-Synthesis Model
- Phases of Compiler
- Lexical Analysis
- Input Buffering
- Specification & Recognition of Tokens
- LEX Tool

🌟 1. INTRODUCTION TO COMPILER

📌 Definition

Compiler ek **system software** hota hai jo:

- High Level Language (HLL)

  ko

- Machine Level Language (MLL)

me convert karta hai.

🎯 Real Life Example

```
int a = 10;
printf("%d",a);
```

Ye program directly computer nahi samajhta.

Compiler isse machine code me convert karta hai:

```
101010101010...
```

Tab computer execute karta hai.

🌟 Why Compiler Needed?

Computer sirf:

```
0 and 1
(Binary Language)
```

samajhta hai.

Hum:

```
C
C++
Java
Python
```

jaise languages me code likhte hain.

Compiler dono ke beech bridge ka kaam karta hai.

🌟 Working of Compiler

```
Source Program
      ↓
   Compiler
      ↓
Object Program
      ↓
 Execution
```

🌟 Features of Compiler

Feature	Explanation
Whole program translation	Ek saath pura program convert karta hai
Faster execution	Machine code fast execute hota hai
Error reporting	Errors compile hone ke baad show karta hai
Object code generation	Executable file banata hai

🌟 Advantages of Compiler

✅ Fast Program Execution

Program ek baar compile hone ke baad fast run hota hai.

✅ Better Optimization

Compiler unnecessary code hata deta hai.

✅ Error Detection

Syntax aur semantic errors detect karta hai.

🚨 PYQ ALERT

🔥 Define Compiler
🔥 Explain working of compiler
🔥 Advantages of compiler

🌟 2. COMPILER VS INTERPRETER

Compiler	Interpreter
Whole program translate karta hai	Line by line translate karta hai
Faster execution	Slow execution
Object code generate karta hai	Object code nahi banata
Errors end me show karta hai	Errors instantly show karta hai
Example: C, C++	Example: Python

🎯 Easy Trick to Remember

```
Compiler → Complete program
Interpreter → Instant line execution
```

🌟 3. STRUCTURE OF COMPILER 🔥

VERY IMPORTANT

📌 Basic Structure

```
Source Program
      ↓
Lexical Analyzer
      ↓
Syntax Analyzer
      ↓
Semantic Analyzer
      ↓
Intermediate Code Generator
      ↓
Code Optimizer
      ↓
Code Generator
      ↓
Target Program
```

🎯 Explanation of Structure

Compiler multiple phases me kaam karta hai.

Har phase ka alag kaam hota hai.

🌟 4. PHASES OF COMPILER 🔥 MOST IMPORTANT 14 MARK QUESTION

📌 Phase 1: Lexical Analysis

🎯 Definition

Source code ko characters se tokens me convert karta hai.

Also called:

```
Scanner
Tokenizer
```

🎯 Example

Program:

```
int a = 10;
```

Tokens:

```
int
a
=
10
;
```

🎯 Functions

✅ Spaces remove karta hai
✅ Comments remove karta hai
✅ Tokens generate karta hai
✅ Lexical errors detect karta hai

📌 Phase 2: Syntax Analysis

🎯 Definition

Program grammar check karta hai.

Also called:

```
Parser
```

🎯 Example

Correct:

```
int a;
```

Wrong:

```
int ;
```

Parser syntax error detect karega.

🎯 Parse Tree

```
   =
  / \
 a  10
```

📌 Phase 3: Semantic Analysis

🎯 Definition

Program ka meaning check karta hai.

🎯 Example

```
int a;
a = "Hello";
```

Error:

```
Type Mismatch
```

🎯 Functions

✅ Type checking
✅ Variable declaration checking
✅ Scope checking

📌 Phase 4: Intermediate Code Generation

Machine independent code generate karta hai.

🎯 Example

```
a = b + c
```

Intermediate code:

```
t1 = b + c
a = t1
```

📌 Phase 5: Code Optimization

Program ko:

- Faster
- Smaller
- Efficient

banata hai.

🎯 Example

Before optimization:

```
a = 2 * 10;
```

After optimization:

```
a = 20;
```

📌 Phase 6: Code Generation

Final machine code generate karta hai.

🌟 Symbol Table

Compiler identifiers ki information store karta hai.

Identifier	Type	Address
a	int	2000

🌟 Error Handler

Har phase me errors detect karta hai.

🚨 PYQ ALERT

🔥 Explain phases of compiler with diagram
🔥 Explain lexical analysis
🔥 Explain parser
🔥 Explain symbol table

🌟 5. ANALYSIS-SYNTHESIS MODEL

📌 Analysis Phase (Front End)

Source program analyze karta hai.

Includes:

- Lexical Analysis
- Syntax Analysis
- Semantic Analysis

📌 Synthesis Phase (Back End)

Target machine code generate karta hai.

Includes:

- Code Optimization
- Code Generation

🎯 Diagram

```
Source Program
      ↓
Analysis Phase
(Front End)
      ↓
Intermediate Code
      ↓
Synthesis Phase
(Back End)
      ↓
Target Program
```

🌟 6. MAJOR DATA STRUCTURES IN COMPILER

Data Structure	Use
Stack	Parsing
Queue	Input buffering
Tree	Parse tree
Graph	Optimization
Symbol Table	Identifier storage

🌟 STACK IN COMPILER

Parsing me use hota hai.

```
PUSH
POP
```

operations perform karta hai.

🌟 Parse Tree

```
   +
  / \
 a   b
```

Expression structure show karta hai.

🌟 7. BOOTSTRAPPING

📌 Definition

Compiler ko compiler ki help se develop karna:

`Bootstrapping`

kehlata hai.

🎯 Example

C language ka compiler:

- C language me hi banana

🌟 Advantages

✅ Faster compiler development
✅ Easy maintenance
✅ Machine independence

🌟 8. PORTING

📌 Definition

Compiler ko ek system se doosre system me transfer karna:

```
Porting
```

kehlata hai.

🎯 Example

```
Windows Compiler
        ↓
Linux Compiler
```

🌟 Advantages

✅ Multiple platforms support
✅ Reusability
✅ Easy migration

🌟 9. LEXICAL ANALYSIS 🔥 VERY IMPORTANT

📌 Definition

Lexical Analyzer:

- Input characters read karta hai
- Tokens generate karta hai

🌟 Important Terms

📌 Token

Meaningful unit.

Example:

```
if
while
+
=
```

📌 Lexeme

Actual character sequence.

Example:

```
count
100
```

📌 Pattern

Rule used to identify token.

Example:

```
digit → 0|1|2|3...
```

🌟 Token Recognition Process

```
Characters
    ↓
Lexical Analyzer
    ↓
Tokens
```

🌟 10. INPUT BUFFERING

📌 Need

Character-by-character reading slow hoti hai.

Buffer use karke:

✅ Fast input possible hota hai.

🌟 Buffer Pair Technique

```
Buffer 1 ↔ Buffer 2
```

Ek buffer process hota hai,

doosra load hota hai.

🌟 11. LEX TOOL

📌 Definition

LEX ek automatic lexical analyzer generator tool hai.

🌟 Working

Input:

```
Regular Expressions
```

Output:

```
Lexical Analyzer Program
```

🌟 Structure of LEX Program

```
Definitions
%%
Rules
%%
User Functions
```

🌟 Example

```
digit [0-9]

%%
{digit}+ printf("NUMBER");
%%
```

🌟 Advantages of LEX

✅ Fast scanner generation

✅ Time saving

✅ Easy implementation

🔥 MOST IMPORTANT 14-MARK QUESTIONS

⭐ Very Very Important

1. Explain phases of compiler with neat diagram.
2. Explain lexical analysis in detail.

3. Explain compiler structure.
4. Explain Analysis-Synthesis model.
5. Explain Bootstrapping and Porting.
6. Explain LEX tool with example.
7. Explain tokens, lexemes and patterns.
8. Explain input buffering.

⚡ Last Minute Revision

✔ Compiler converts HLL to MLL
✔ Lexical Analyzer generates tokens
✔ Parser checks grammar
✔ Semantic Analyzer checks meaning
✔ Symbol table stores identifiers
✔ Bootstrapping = compiler using compiler
✔ Porting = transfer compiler to another system
✔ LEX generates lexical analyzer
✔ Token recognition uses Regular Expressions
✔ Input buffering increases speed