Compiler Design and Construction

Based on Hardware:

1. Analysis Phase:
    1. Lexical Analysis
    2. Syntax Analysis (Recursive Descent)
    3. Semantic Analysis
2. Synthesis Phase:
    1. Code Generation/Intermediate Language
    2. Optimization
    3. Target Code Generation
3. Symbol Table
4. Error Handling
5. Others:
    - Exception Handling
    - Linking & Loading
    - Debugging Tools
    - Profiling Tools

Based on Software:

1. Frontend:
    1. Lexical Analysis
    2. Syntax Analysis (Recursive Descent)
    3. Semantic Analysis
    4. Intermediate Code Generation
2. Backend:
    1. Optimization
    2. Target Code Generation

1. Lexical Analysis

- Input Streams(Lexem), Tokens, Token Types
- Regular Expressions for Pattern Matching
- Case Insensitivity vs Case Sensitivity
- Keywords vs Identifiers
- Multi-Character Operators
- White Space Characters
- Comments

2. Syntax Analysis (Recursive Descension)

- Recursion Trees / Parse Tables
- Left Recursion Resolution
- Right Recursion Resolution
- Conflict Resolution Techniques
- LL(k) Grammars
- Context Free Languages

3. Semantic Analysis

- Augmented Parse Tree
- Type Checking
- Scoping Rules
- Name Binding
- Overloading Resolution
- Conversion between Data types
- Evaluation of expressions

4. Code Generation/ Intermediate Language

- Target Machine Architecture
- Instructions Set
- Register Allocation
- Addressing Modes
- Intermediate Language Formats
- Translation to the target machine code

5. Optimization

- Basic Block Propagation
- Dead Store Elimination
- Constant Folding and Propagation
- Common Subexpression Elimination
- Strength Reduction
- Loop Invariant Code Motion
- Dead Loop Detection
- Function Inlining
- Jump Threading
- Live Variable Analysis
- Escape Analysis
- Profile Guided Optimizations
- Static Single Assignment Form (SSA)
- Control Flow Simplification
- Code Sparsing
- Opaque Pointers
- Link Time Optimization (LTO)

Symbol Table

Data Structure

- Variables
- Function names
- Classes
- Interfaces

Legical

- Creates Entries for identifiers

Syntax:

- Adds information regarding attributes

Semantics:

- Using available info checks seman and updates symbol table

Intermediate C.G.

- Available info helps in adding temporary variables

Code optimization:

- USed in machine dependent optimization

Taeget Code Generation

- Generetes target code using address of Identifiers

Symbol Table

```
Assume each address block in memory is of 8 bits(1 byte)

1. .......
2. .......
3. int n -------------(i)
4. float a[5];

Linked List Representation:
|3| |--->|10 | |--->|12| |
```

S.No.	Type	Size	Dimension	Line of Declaration	Line of Usage	Address
1	int	2 bytes	0	3	10,12	2005H
2	float	4 bytes	1	4	15,16,17	2006H