MADE EASY
COMPUTER SCIENCE
Topper Handwritten Notes
COMPILER DESIGN
BY-PRASAD SIR

- Theory
- Explanation
- Derivation
- Example
- Shortcuts
- Previous Years Question With Solution

Compiler Design

Topics List:

30 hrs. ≃ 12-15 days.

- Basics of a compiler
- Lexical Analysis
- Syntax Analysis
- Syntax Directed Translation
- Intermediate Code Generator
- Code Optimization
- Run Time Environment

Text Book:

Compilers, Techniques & Tools
By Ullman

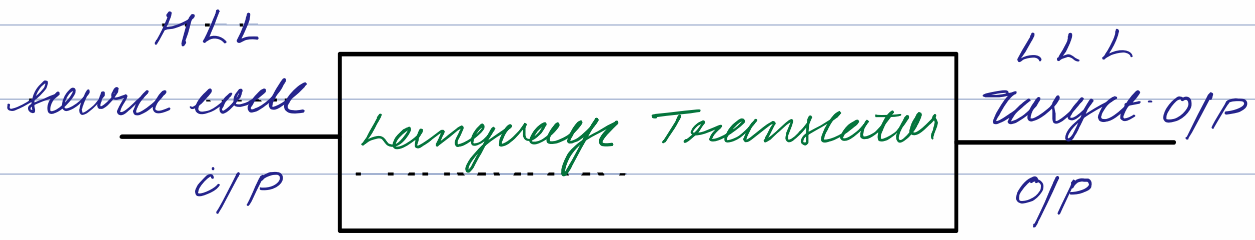
Marks: 4 to 9.

8500518598
prasad Sir (TOC & CD)

L#1 18/08/23

Basics of a compiler

Language Translator : A Language Translator takes one language as input and produces another language as output.



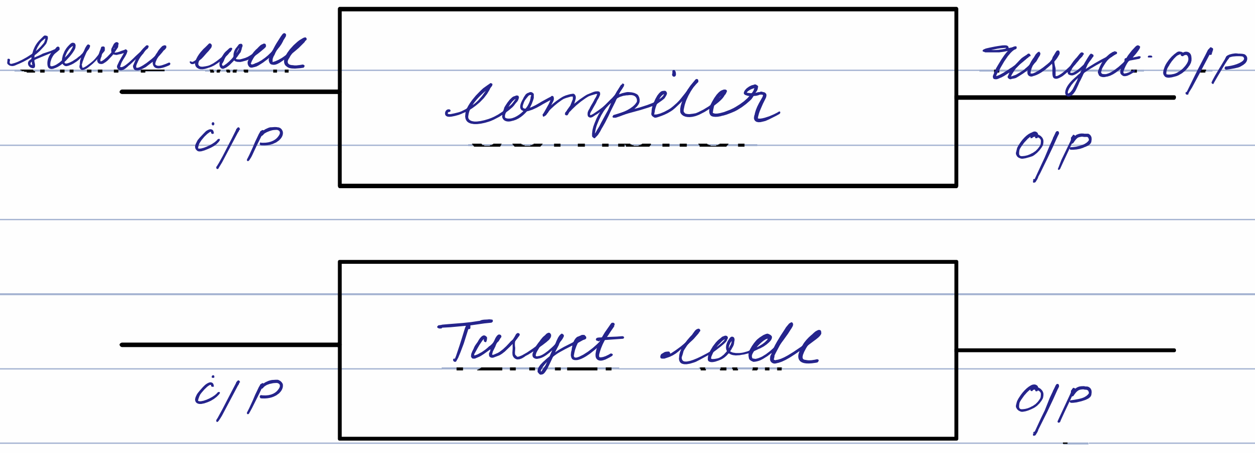
Types of Language translators

- compiler
- Assembler
- Interpreter

1. **compiler** : compiler takes the source code as input and produces the target code as output. If this target code is an executable code then it'll be called by the user to provide the inputs for producing the output.

compiler executes the entire code at a time from top to bottom. If any error present at any line all of them will be given

Error Diagnosis in compiler is difficult compared to the interpreter but output generation by compiler is faster. compilation is an offline process.

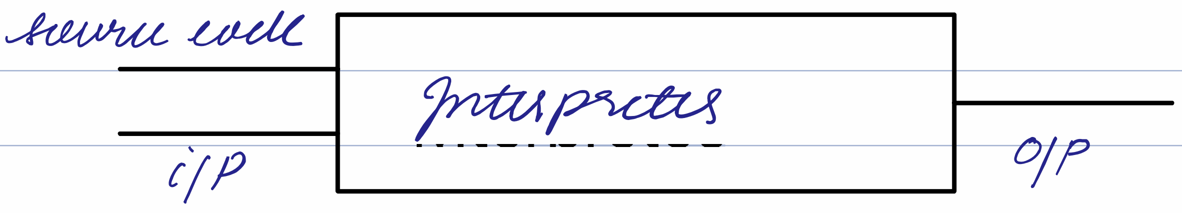


Ex: Pascal, C, C++, C#

2. Interpreter: Interpreter takes the source code as input and produces the direct output. It will not produce any intermediate language as in the case of compiler.

Interpreter executes the source code line by line, if any error present at any line immediately that error will be produc-ed. Untill the programmer resolves that error the interpreter will not execute the next line.

Error Diagnosis is easy in the case of Interpreter.
The Interpreter executes the each statement and it process the inputs simultaneously.
Thus, interpreter is Online process.
As interpreter produces the output directly we need not to store the executable code anywhere in the main memory.
Thus, Interpreter takes less memory compared to the compiler. The end user can easily modify the source program in the case of interpreter.



Ex: python, LISP, PERL, RUBY

3. Assembler: Assembler takes Assembly language code as input and produces relocatable machine code as output which is ready for execution.
Assembly language uses opcode for the instructions. An opcode basically gives the information about the particular instruction. The symbolic representation of the opcode is called as Mnemonics.