VIGNAN'S NIRULA
INSTITUTE OF TECHNOLOGY & SCIENCE FOR WOMEN
(Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada)
Vignan Avenue, Peda Palakaluru, Guntur - 522009

Department of Information Technology

Innovative Teaching Practice

Faculty Name	**:**	Dr. K. V. S. S. Rama Krishna, Dr. K. Venkateswara Rao
Course Name	:	Automata Theory and Compiler Design
Class	**:**	II B. Tech II Semester
Academic Year	**:**	2021-2022
Title of the Topic	**:**	Code Generation
Activity Name	**:**	Think pair share

Objective:

To promote active participation, collaboration, and deeper understanding. It encourages individuals to first think independently about a question or topic, then discuss their thoughts with a partner, and finally share their insights with the larger group. This method enhances learning by allowing time for reflection, peer interaction, and collective idea exchange, all in a structured, efficient way.

Method to Implement:

1. Pre-Class Learning (Online Phase)

- **Pre-recorded Lectures:** Short videos explaining the concepts of code generation, parsing techniques, syntax trees, and intermediate representations.
- **Reading Material:** Chapters or articles covering code generation principles, optimization techniques, and the role of code generators in compiler design, with real-world examples.

2. In-Class Learning (Face-to-Face Phase)

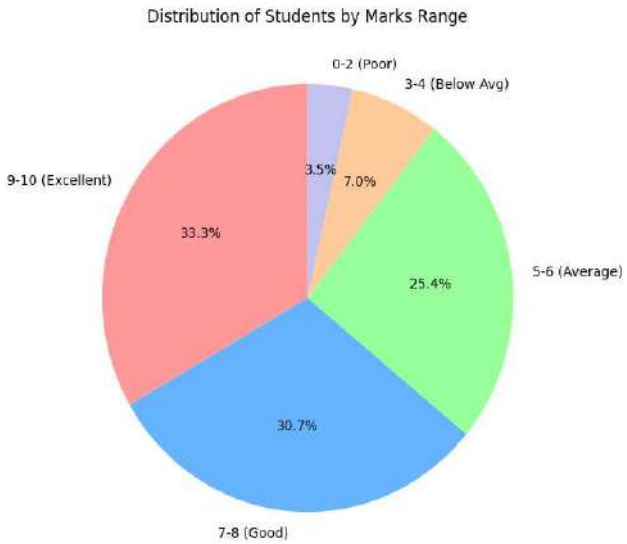
- **Discussion Sessions:** Recap the online materials with discussions on how code generators work and their applications in compilers and other automated systems.
- **Hands-on Exercises:** Practice generating code using tools like YACC or Bison, or through simple code generator projects using languages like C, Java, or Python.

3. Post-Class Learning (Reinforcement Phase)

- **Follow-up Videos:** Additional content on advanced topics like Just-in-Time (JIT) compilers, optimization during code generation, and the design of code generation backends.
- **Assignments:** Solve practical problems related to code generation, like writing a simple code generator for an intermediate language or optimizing generated code.

Screenshot of the Practice

Marks Range	Number of Students	Percentage
9-10 (Excellent)	38	33.33%
7-8 (Good)	35	30.70%
5-6 (Average)	29	25.44%
3-4 (Below Avg)	8	7.02%
0-2 (Poor)	4	3.51%
Total	**114**	100%



Conclusion

The code generator is a critical component of the compiler, responsible for translating intermediate representations into efficient machine code. Optimizations like constant propagation, partial-redundancy elimination, and loop transformations significantly influence the quality of the generated code. By leveraging these techniques, the code generator ensures the output program is both performant and portable across different machine architectures. This process is integral to producing high-quality executables that maximize hardware utilization while adhering to language semantics.

Signature of the Faculty **Head of the Department**