**Q3 . DATA Flow :-**

In compiler construction, data flow refers to the movement and transformation of data through a program during its execution. Analyzing data flow is crucial for optimizing code and understanding how variables are used and modified throughout the program. Code :- class Compiler: def \_\_init\_\_(self): self.variable\_table = {} def compile\_assignment(self, variable, expression): # Compile the expression self.compile(expression) # Emit code to store the result in the variable self.emit("STORE", variable) def compile\_arithmetic\_operation(self, operand1, operand2, operator, result\_variable): # Compile operands self.compile(operand1) self.compile(operand2) # Emit code for the arithmetic operation if operator == '+': self.emit("ADD") elif operator == '-': self.emit("SUB") elif operator == '\*': self.emit("MUL") elif operator == '/': self.emit("DIV") # Store the result in the specified variable self.emit("STORE", result\_variable) def compile(self, code): # Simplified compilation of code print("Compiling:", code) def emit(self, operation, operand=None): print(f"{operation} {operand}") # Example usage compiler = Compiler() # Example 1: Simple assignment assignment\_code = "x 10" compiler.compile\_assignment("x", "10") # Example 2: Arithmetic operation arithmetic\_code = "y x 5 +" compiler.compile\_arithmetic\_operation("x", "5", "+", "y")