**Name: Ayush Kushwaha**

**Roll No:** CH.EN.U4CSE22077

**Lab-9**

**Aim:** To implement target code generation

**Algorithm:**

* Read input string
* Consider each input string and convert it to machine code instructions using switch case
* Load the input variables into new variables as operands and display them using “load”
* With the help of arithmetic operation, we will display arithmetic operations like add, sub, div, mul for the respective operations in switch case
* Generate 3 address code for each input variable.
* If ‘=‘ is seen as arithmetic operation, then store the result in a variable and display it with “store”.
* Repeat this for each line in the input string.
* Display the output which gives a transformed input string of assembly language code.

**Code:**

def generate\_target\_code(intermediate\_code): target\_code = [] reg\_count = 1

for stmt in intermediate\_code:

left, right = stmt.split("=") left = left.strip()

right = right.strip()

if any(op in right for op in ["+", "-", "\*", "/"]): tokens = right.split()

op1, operator, op2 = tokens[0], tokens[1], tokens[2]

**Name: Ayush Kushwaha**

**Roll No:** CH.EN.U4CSE22077

target\_code.append(f"LOAD R{reg\_count}, {op1}") target\_code.append(f"{operator\_map(operator)} R{reg\_count}, {op2}") target\_code.append(f"STORE {left}, R{reg\_count}")

else:

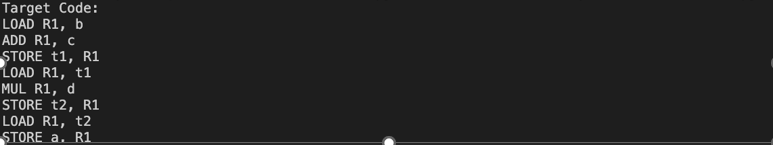
target\_code.append(f"LOAD R{reg\_count}, {right}") target\_code.append(f"STORE {left}, R{reg\_count}")

return target\_code

def operator\_map(op): return { "+": "ADD", "-": "SUB", "\*": "MUL", "/": "DIV" }[op] intermediate = [ "t1 = b + c", "t2 = t1 \* d", "a = t2" ]

result=generate\_target\_code(intermediate) print("Target Code:") for line in result: print(line)

**Output:**

****

**Result:**

The program to implement target code generation has been successfully executed.