**Shri Ramdeobaba College of Engineering and Management, Nagpur**

**Department of Computer Science and Engineering**

**Session: 2024-25**

**Compiler Design Lab**

**PRACTICAL No. 8**

**Topic:** Code Generation

**Platform:** Windows or Linux

**Language to be used:** Python or Java (based on the companies targeted for placement)

**Aim:** Write a program to generate the code using **simple code generation algorithm**

1. **Create appropriate data structures for Register descriptor and Address descriptor.**
2. **Implement getreg ( ) function by checking the created data structures.**
3. **Input the three address code and generate the target code.**

**Code :**

REGISTERS = ["R0", "R1", "R2", "R3"]

reg\_descriptor = {reg: None for reg in REGISTERS}

addr\_descriptor = {}

def getreg(var=None):

for reg, val in reg\_descriptor.items():

if val is None:

return reg

if var:

for reg, val in reg\_descriptor.items():

if val == var:

return reg

reg\_to\_spill = REGISTERS[0]

spilled\_var = reg\_descriptor[reg\_to\_spill]

if spilled\_var:

addr\_descriptor[spilled\_var] = 'memory'

print(

f"MOV [{spilled\_var}], {reg\_to\_spill} ; Spill {spilled\_var} to memory"

)

reg\_descriptor[reg\_to\_spill] = None

return reg\_to\_spill

def generate\_code(tac\_list):

for stmt in tac\_list:

parts = stmt.split()

if len(parts) == 5:

dest, \_, op1, operator, op2 = parts

reg1 = getreg(op1)

if addr\_descriptor.get(op1) != reg1:

print(f"MOV {reg1}, [{op1}]")

addr\_descriptor[op1] = reg1

reg2 = getreg(op2)

if addr\_descriptor.get(op2) != reg2:

print(f"MOV {reg2}, [{op2}]")

addr\_descriptor[op2] = reg2

result\_reg = reg1

if operator == '+':

print(f"ADD {result\_reg}, {reg2}")

elif operator == '-':

print(f"SUB {result\_reg}, {reg2}")

elif operator == '\*':

print(f"MUL {result\_reg}, {reg2}")

elif operator == '/':

print(f"DIV {result\_reg}, {reg2}")

reg\_descriptor[result\_reg] = dest

addr\_descriptor[dest] = result\_reg

elif len(parts) == 3:

dest, \_, src = parts

reg = getreg(src)

if addr\_descriptor.get(src) != reg:

print(f"MOV {reg}, [{src}]")

addr\_descriptor[src] = reg

print(f"MOV [{dest}], {reg}")

reg\_descriptor[reg] = dest

addr\_descriptor[dest] = 'memory'

tac\_code = ["t1 = a + b", "t2 = t1 \* c", "t3 = t2 - d", "x = t3"]

generate\_code(tac\_code)

Output :

