class AssemblerPassOne:

def \_\_init\_\_(self):

self.symbol\_table = {}

self.intermediate\_code = []

self.opcode\_table = {

"ADD": "01", "SUB": "02", "MUL": "03", "DIV": "04",

"LOAD": "05", "STORE": "06", "START": "00", "END": "FF"

}

self.current\_address = 0

def pass\_one(self, source\_code):

for line in source\_code:

parts = line.strip().split()

if not parts:

continue

if parts[0] == "START":

self.current\_address = int(parts[1]) if len(parts) > 1 else self.current\_address

continue

if parts[0] == "END":

self.intermediate\_code.append((self.current\_address, "END"))

break

if parts[0].endswith(":"):

self.symbol\_table[parts[0][:-1]] = self.current\_address

parts = parts[1:]

if parts and parts[0] in self.opcode\_table:

self.intermediate\_code.append((self.current\_address, parts[0]))

self.current\_address += 1

elif len(parts) >= 2 and parts[1] == "=":

self.symbol\_table[parts[0]] = self.current\_address

self.current\_address += 1

def display\_tables(self):

print("Symbol Table:", self.symbol\_table)

print("Intermediate Code:")

for address, instruction in self.intermediate\_code:

print(f"{address}: {instruction}")

# Sample input assembly code

source\_code = [

"START 100",

"LOAD A",

"ADD B",

"STORE RESULT",

"END",

"A = 5",

"B = 10",

"RESULT ="

]

assembler = AssemblerPassOne()

assembler.pass\_one(source\_code)

assembler.display\_tables()

class AssemblerPassTwo:

def \_\_init\_\_(self):

self.opcode\_table = {

"ADD": "01", "SUB": "02", "MUL": "03", "DIV": "04",

"LOAD": "05", "STORE": "06", "START": "00", "END": "FF"

}

self.current\_address = 0

def pass\_two(self, intermediate\_code, symbol\_table, literals, pool\_table):

machine\_code = []

for address, instruction in intermediate\_code:

if instruction in self.opcode\_table:

machine\_code.append((address, self.opcode\_table[instruction]))

elif instruction == "END":

machine\_code.append((address, "FF"))

elif instruction in symbol\_table:

machine\_code.append((address, str(symbol\_table[instruction])))

elif instruction in literals:

machine\_code.append((address, instruction))

for index in pool\_table:

machine\_code.append((self.current\_address, literals[index]))

self.current\_address += 1

return machine\_code

def display\_machine\_code(self, machine\_code):

print("\nMachine Code:")

for address, code in machine\_code:

print(f"{address}: {code}")

# Example usage

symbol\_table = {"A": 104, "B": 105, "RESULT": 106}

intermediate\_code = [

(100, "LOAD"), (101, "ADD"), (102, "STORE"), (103, "END"),

(104, "A"), (105, "B"), (106, "RESULT")

]

literals = ["5", "10"]

pool\_table = [0, 1]

assembler\_pass\_two = AssemblerPassTwo()

machine\_code = assembler\_pass\_two.pass\_two(intermediate\_code, symbol\_table, literals, pool\_table)

assembler\_pass\_two.display\_machine\_code(machine\_code)