Name : Khune Radha Rahul

Class : TE Comp-2

Roll no : 07(F22112010)

**Input:**

class Assembler:

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.literals = []

self.pool\_table = []

self.current\_address = 0

self.literal\_index = 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

# Check for labels and add to symbol table

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

label = parts[0][:-1]

self.symbol\_table[label] = self.current\_address

parts = parts[1:] # Remove the label

if parts:

opcode = parts[0]

if opcode in self.opcode\_table:

self.intermediate\_code.append((self.current\_address, opcode))

self.current\_address += 1 # Increment address for each instruction

# Handle literals

for part in parts:

if part.startswith('='):

literal = part[1:] # Remove '='

if literal not in self.literals:

self.literals.append(literal)

self.pool\_table.append(self.literal\_index) # Add index of first literal

self.literal\_index += 1

def pass\_two(self):

machine\_code = []

for code in self.intermediate\_code:

address = code[0]

instruction = code[1]

if instruction in self.opcode\_table:

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

elif instruction == "END":

machine\_code.append((address, "FF")) # End opcode

# Handle literals based on pool table

for index in self.pool\_table:

if index < len(self.literals):

literal = self.literals[index]

address = self.current\_address

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

self.current\_address += 1

return machine\_code

def display\_tables(self):

print("Symbol Table:")

for label, address in self.symbol\_table.items():

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

print("\nLiteral Table:")

for i, literal in enumerate(self.literals):

print(f"Literal {i + 1}: {literal}")

print("\nPool Table:")

for i, index in enumerate(self.pool\_table):

print(f"Pool {i + 1}: Index {index}")

print("\nIntermediate Code:")

for code in self.intermediate\_code:

print(code)

# Read from the .asm file

def main():

assembler = Assembler()

with open('sample.asm', 'r') as file:

source\_code = file.readlines()

assembler.pass\_one(source\_code)

assembler.display\_tables()

machine\_code = assembler.pass\_two()

print("\nMachine Code:")

for address, code in machine\_code:

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

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Input Assembly language file:**

START 200

LOAD A

ADD B

STORE RESULT

END

A = 5

B = 10

RESULT =

**Output:**

admin1@linux:~$ python3 assembler.py

Symbol Table:

RESULT: 203

A: 204

B: 205

Literal Table:

Literal 1: 5

Literal 2: 10

Pool Table:

Pool 1: Index 0

Intermediate Code:

(200, 'LOAD')

(201, 'ADD')

(202, 'STORE')

(203, 'END')

Machine Code:

200: 05

201: 01

202: 06

203: FF

204: 5

205: 10