(A)4.Write a program to generate Intermediate code of a two-pass Assembler for the given Assembly

language source code.

INPUT/CODE

START 100

READ A

READ B

MOVER AREG, A

SUB AREG, B

STOP

A DS 1

B DS 1

END  
  
  
class IntermediateCodeGenerator:

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

self.intermediate\_code = []

def generate\_intermediate\_code(self, source\_code):

for line in source\_code:

tokens = line.split()

opcode = tokens[0]

operand = ""

if len(tokens) > 1:

operand = tokens[1]

if opcode == "START":

self.intermediate\_code.append("AD " + opcode + ", " + operand)

elif opcode == "READ":

self.intermediate\_code.append("IS 1, " + operand)

elif opcode == "MOVER":

self.intermediate\_code.append("IS 4, " + operand + "AREG")

elif opcode == "SUB":

self.intermediate\_code.append("IS 2, " + operand + "BREG")

elif opcode == "STOP":

self.intermediate\_code.append("IS 0")

elif opcode == "DS":

self.intermediate\_code.append("DL 1, " + operand)

elif opcode == "END":

self.intermediate\_code.append("AD " + opcode)

else:

# Ignore comments and unrecognized instructions

pass

def print\_intermediate\_code(self):

print("Intermediate Code:")

for code in self.intermediate\_code:

print(code)

source\_code = [

"START 100",

"READ A",

"READ B",

"MOVER AREG, A",

"SUB AREG, B",

"STOP",

"A DS 1",

"B DS 1",

"END"

]

intermediate\_generator = IntermediateCodeGenerator()

intermediate\_generator.generate\_intermediate\_code(source\_code)

intermediate\_generator.print\_intermediate\_code()