(A)2. Write a program to generate Literal table of a two-pass Assembler for the given Assembly

language source code.

INPUT/CODE

START 100

READ A

READ B

MOVER AREG, =’50&#39;

MOVER BREG, =’60’

ADD AREG, BREG

LOOP MOVER CREG, A

ADD CREG, =&#39;10&#39;

COMP CREG, B

BC LT, LOOP

NEXT SUB AREG, =&#39;10&#39;

COMP AREG, B

BC GT, NEXT

STOP

A DS 1

B DS 1

END  
  
def generate\_literal\_table(source\_code):

literal\_table = {}

lc = 100 # Location Counter starts at 100

for line in source\_code:

if line.strip() == '':

continue

tokens = line.split()

opcode = tokens[0]

if opcode == 'START':

lc = int(tokens[1])

elif opcode == 'READ' or opcode == 'MOVER' or opcode == 'MOVEM' or opcode == 'ADD' or opcode == 'SUB' or opcode == 'COMP':

for token in tokens[1:]:

if token.startswith('='):

literal = token[2:-1]

if literal not in literal\_table:

literal\_table[literal] = lc

# Don't increment lc here, it's already assigned for the literal

else:

lc += 1 # Increment lc only for instructions without literals

elif opcode == 'END':

break

else:

lc += 1 # Increment lc for non-literal instructions

return literal\_table

source\_code = [

"START 100",

"READ A",

"READ B",

"MOVER AREG, ='50'",

"MOVER BREG, ='60'",

"ADD AREG, BREG",

"LOOP MOVER CREG, A",

"ADD CREG, ='10'",

"COMP CREG, B",

"BC LT, LOOP",

"NEXT SUB AREG, ='10'",

"COMP AREG, B",

"BC GT, NEXT",

"STOP",

"A DS 1",

"B DS 1",

"END"

]

literal\_table = generate\_literal\_table(source\_code)

print("Literal Table:")

print("Literal\t\tAddress")

for literal, address in literal\_table.items():

print(f"{literal}\t\t{address}")