(A)3.Write a program to generate Pool table of a two-pass Assembler for the given Assembly language

source code.

INPUT/CODE

START 100

READ A

MOVER AREG, =&#39;1&#39;

MOVEM AREG, B

MOVER BREG, =&#39;6&#39;

ADD AREG, BREG

COMP AREG, A

BC GT, LAST

LTORG

NEXT SUB AREG, =&#39;1&#39;

MOVER CREG, B

ADD CREG, =&#39;8&#39;

MOVEM CREG, B

PRINT B

LAST STOP

A DS 1

B DS 1

END  
  
  
def generate\_pool\_table(assembly\_code):

pool\_table = []

current\_pool\_start = None

# Split assembly code into lines

lines = assembly\_code.strip().split('\n')

# Process each line of the assembly code

for line in lines:

tokens = line.split()

# Check if the line has tokens

if len(tokens) > 0:

mnemonic = tokens[0]

# Check for LTORG directive to identify the end of a literal pool

if mnemonic == "LTORG":

if current\_pool\_start is not None:

# End of current pool, record the pool start address

pool\_table.append(current\_pool\_start)

current\_pool\_start = None

elif mnemonic in ['MOVER', 'MOVEM', 'ADD', 'COMP', 'SUB', 'PRINT']:

for operand in tokens[1:]:

if operand.startswith("='") and operand.endswith("'"):

# Literal found, check if it's the start of a new pool

if current\_pool\_start is None:

current\_pool\_start = len(pool\_table) # Use index as address for simplicity

# Check if there's an open pool at the end of the code

if current\_pool\_start is not None:

pool\_table.append(current\_pool\_start)

return pool\_table

# Example Assembly Language Code

assembly\_code = """

START 100

READ A

MOVER AREG, ='1'

MOVEM AREG, B

MOVER BREG, ='6'

ADD AREG, BREG

COMP AREG, A

BC GT, LAST

LTORG

NEXT SUB AREG, ='1' MOVER CREG, B

ADD CREG, ='8'

MOVEM CREG, B

PRINT B

LAST STOP

A DS 1

B DS 1

END

"""

# Generate pool table from assembly code

pool\_table = generate\_pool\_table(assembly\_code)

# Print pool table in the expected format

print("-------------")

print("Pool Address")

print("-------------")

for i, address in enumerate(pool\_table):

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