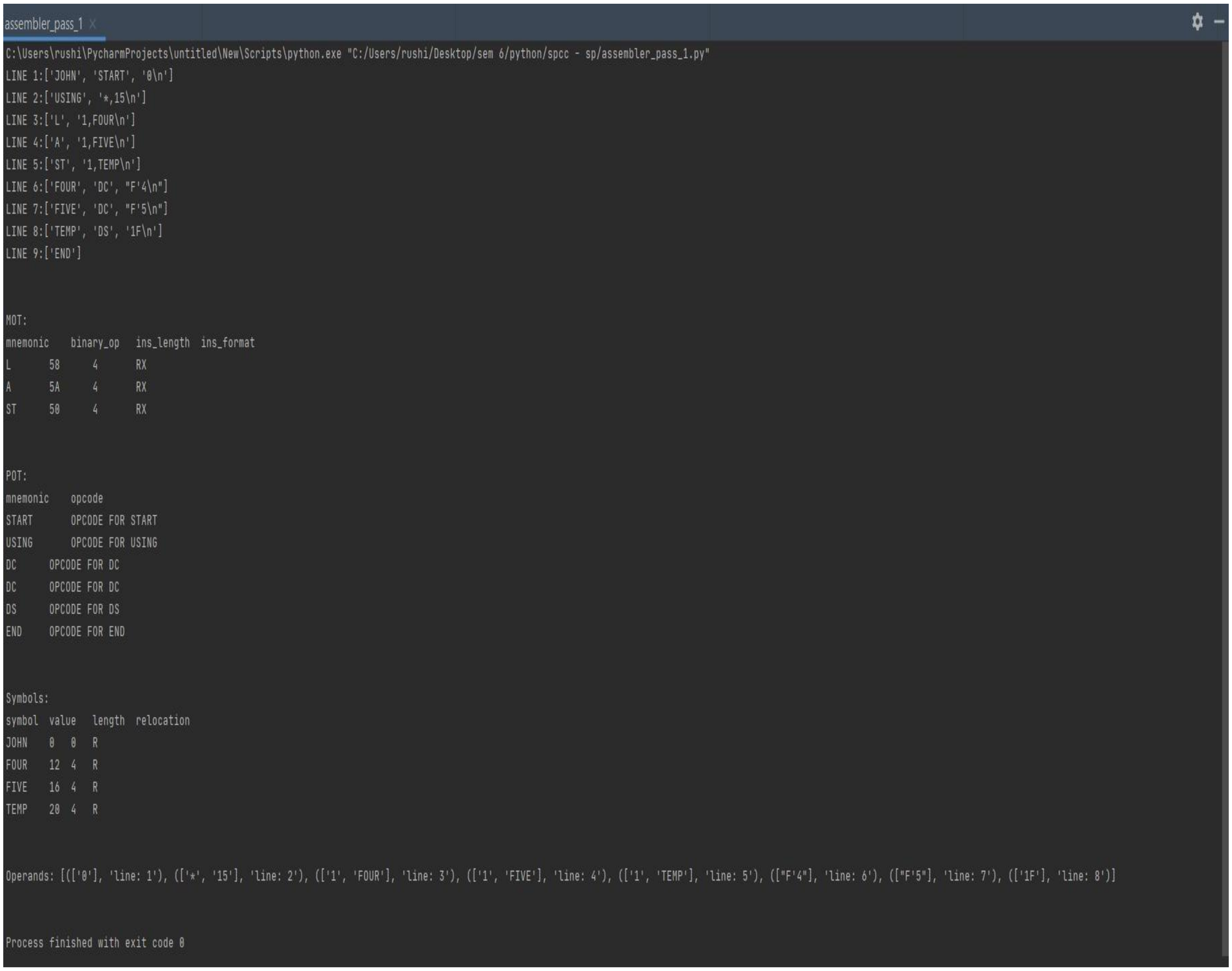
PASS 1 ASSEMBLER

```
code = []

MOT_ref={
  'L': ['L','58','4','RX'],
  'A': ['A','5A','4','RX'],
  'ST': ['ST','50','4','RX'],
  'BASR':['BASR','0D','4','RX'],
  'BALR':['BALR','05','2','RR'],
}

POT_ref={
  'START': ['START','OPCODE FOR START'],
  'USING': ['USING','OPCODE FOR USING'],
  'END': ['END', 'OPCODE FOR END'],
  'DC': ['DC','OPCODE FOR DC'],
  'DS': ['DS', 'OPCODE FOR DS']
}
lc = 0
length = 0
MOT = [] POT = []
symbolTable = [] operands
= [] # take input form file
with open('./textfile.txt') as
f: code = f.readlines()

for i,line in enumerate(code):
  tokens = line.split('   ')
  print(f"LINE
  {i+1}:{tokens}") for token
  in tokens: if token in
  MOT_ref:
        length = int(MOT_ref[token][2])
        lc+=length
        MOT.append(MOT_ref[token])
     elif token in POT_ref:
        if token == "DC" or token ==
           'DS': lc+=4
        POT.append(POT_ref[token])
     else:
        flag = True for
        char in token:
           if char == ',' or char in '0123456789':
              flag = False break if flag:
        symbolTable.append(tuple((token,lc,length,'R')))
        else:
       operands.append(tuple((token[:-1].split(','),f"line: {i+1}")))

print('\n\nMOT:')
print("mnemonic\tbinary_op\tins_length\tins_format")

for x in MOT:
  print(f"{x[0]}\t\t{x[1]}\t\t{x[2]}\t\t{x[3]}")

print('\n\nPOT:')
print("mnemonic\topcode")
for x in POT:
  print(f"{x[0]}\t\t{x[1]}")

print('\n\nSymbols:')
print('symbol\tvalue\tlength\trelocation')
for x in symbolTable:
  print(f"{x[0]}\t{x[1]}\t{x[2]}\t{x[3]}")
print('\n\nOperands:',operands)
print('')
```

textfile.txt

```
          JOHN   START   0
          USING   *,15
          L   1,FOUR
          A   1,FIVE
          ST   1,TEMP
          FOUR   DC   F'4
          FIVE   DC   F'5
          TEMP   DS   1F
          END
```

Output:



```
C:\Users\rushi\PycharmProjects\untitled\New\Scripts\python.exe "C:/Users/rushi/Desktop/sem 6/python/spcc - sp/assembler_pass_1.py"
LINE 1:['JOHN', 'START', '0\n']
LINE 2:['USING', '*,15\n']
LINE 3:['L', '1,FOUR\n']
LINE 4:['A', '1,FIVE\n']
LINE 5:['ST', '1,TEMP\n']
LINE 6:['FOUR', 'DC', "F'4\n"]
LINE 7:['FIVE', 'DC', "F'5\n"]
LINE 8:['TEMP', 'DS', '1F\n']
LINE 9:['END']


MOT:
mnemonic    binary_op   ins_length  ins_format
L       58      4       RX
A       5A      4       RX
ST      50      4       RX


POT:
mnemonic    opcode
START       OPCODE FOR START
USING       OPCODE FOR USING
DC      OPCODE FOR DC
DC      OPCODE FOR DC
DS      OPCODE FOR DS
END     OPCODE FOR END


Symbols:
symbol  value   length  relocation
JOHN    0   0   R
FOUR    12  4   R
FIVE    16  4   R
TEMP    20  4   R


Operands: [(['0'], 'line: 1'), (['*', '15'], 'line: 2'), (['1', 'FOUR'], 'line: 3'), (['1', 'FIVE'], 'line: 4'), (['1', 'TEMP'], 'line: 5'), (["F'4"], 'line: 6'), (["F'5"], 'line: 7'), (['1F'], 'line: 8')]


Process finished with exit code 0
```