Experiment no 2

```
with open('EXP.txt') as t:
   data = []
   for line in t.readlines():
      data.append(line.split())
symbols = []
value = 0
def contains(string):
   string = list(string)
   for i in string:
      if i == "F":
         return 4
      elif i == "D":
         return 8
   return 1
def contains_literal(string):
   string = list(string)
   if "=" in string:
      return True
for j, i in enumerate(data):
   if len(i) == 2 and i[0].lower() == "using":
      value = 0
      continue
   if len(i) == 2:
      value += 4
   if j == 1:
      value = 0
      continue
   if len(i) == 3:
      length = contains(i[2])
      if i[1].lower() == "eqv":
         symbols.append([i[0], int(i[2]), length, 'A'])
         base = int(i[2])
      else:
         symbols.append([i[0], value, length, "R"])
         if (length != 4):
            value += length
         else:
            value += 4
print("OUTPUT of Pass 1\n\nSymbol Table (ST)")
print("Symbol\tValue\t\tLength\tRelocation")
for i in symbols:
   print(i[0], "\t", hex(i[1])[2:], '(', i[1], ')', "\t\t", i[2], "\t", i[3])
literals = []
lvalue = value
for j, i in enumerate(data):
   if len(i) == 2:
      if contains_literal(i[1]):
         a = list((i[1].split('='))[1])
         length = contains(a[0])
         literals.append([(i[1].split(','))[1], lvalue, length, "R"])
```

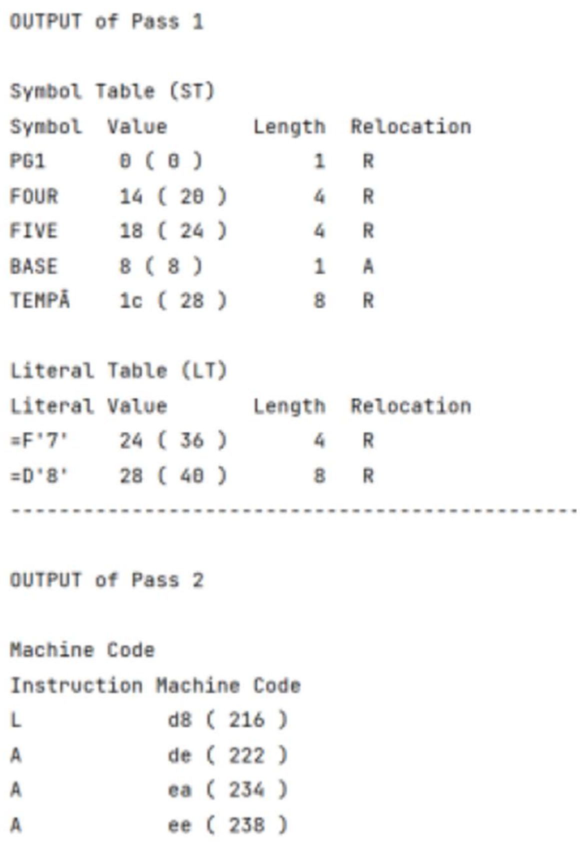
```
            if (length != 4):
                lvalue += length
            else:
                lvalue += 4
print("\nLiteral Table (LT)")
print("Literal\tValue\t\tLength\tRelocation")
for i in literals:
    print(i[0], "\t", hex(i[1])[2:], '(', i[1], ')', "\t\t", i[2], "\t", i[3])
main = symbols + literals
mot = [['L', int('58', 16)], ['ST', int('50', 16)], ['A', int('5A', 16)]]
def getOpHex(op):
    for i in mot:
        if i[0] == op:
            return i[1]
    return
def getOpOperand(op):
    for i in main:
        if i[0] == op:
            return i[1]
    return
print(" ")
print("\nOUTPUT of Pass 2\n\nMachine Code")
print("Instruction\tMachine Code")
one = 100
for i, j in enumerate(data[2:], 1):
    if len(j) == 2:
        final = getOpHex(j[0]) + getOpOperand(j[1].split(',')[1]) + one + base
        print(j[0], '\t\t\t', hex(final)[2:], '(', final, ')')
bases = []
for i in range(0, 16):
    if (i == base):
        bases.append(['Y', 000000])
    else:
        bases.append(['N', None])
print("\nBase Table (BT)")
print("Base Availability Indicator Contents")
for j, i in enumerate(bases):
    if (i[1] == 0):
        print(j, "\t", i[0], "\t\t\t\t\t\t", str(i[1]) * 6)
    else:
        print(j, "\t", i[0])
```

Input:

Exp.txt

```
PG1 START 0
USING *,BASE
L 1,FOUR
A 1,FIVE
A 1,=F'7'
A 1,=D'8'
ST 1,TEMP
FOUR DC F'4'
FIVE DC F'5'
BASE EQV 8
TEMP DC '1'D
END
```

Output:



```
OUTPUT of Pass 1

Symbol Table (ST)
Symbol  Value       Length  Relocation
PG1      0 ( 0 )         1   R
FOUR     14 ( 20 )       4   R
FIVE     18 ( 24 )       4   R
BASE     8 ( 8 )         1   A
TEMPĀ    1c ( 28 )       8   R

Literal Table (LT)
Literal Value       Length  Relocation
=F'7'    24 ( 36 )       4   R
=D'8'    28 ( 40 )       8   R
----------------------------------------------

OUTPUT of Pass 2

Machine Code
Instruction Machine Code
L            d8 ( 216 )
A            de ( 222 )
A            ea ( 234 )
A            ee ( 238 )
```