CODE :

```
from sys import exit
motOpCode = {
    "MOV": 1,
    "A": 2,
    "S": 3,
    "M": 4,
    "D": 5,
    "AN": 6,
    "O": 7,
    "ADD": 8,
    "SUB": 9,
    "MUL": 10,
    "DIV": 11,
    "AND": 12,
    "OR": 13,
    "LOAD": 14,
    "STORE": 15,
    "DCR": 16,
    "INC": 17,
    "JMP": 18,
    "JNZ": 19,
    "HALT": 2}
motSize = {
    "MOV": 1,
    "A": 1,
    "S": 1,
    "M": 1,
    "D": 1,
    "AN": 1,
    "O": 1,
    "ADD": 1,
    "SUB": 2,
    "MUL": 2,
    "DIV": 2,
    "AND": 2,
    "OR ": 2,
    "LOAD": 3,
    "STORE": 3,
    "DCR": 1,
```

```
    "INC": 1,
    "JMP": 3,
    "JNZ": 3,
    "HALT": 1}
l = []
relativeAddress = []
machineCode = []
RA = int(input("Enter the starting address: "))
current = 0
count = 0
print("Enter ALP :")
while True:
  instructions = input()
  l.append(instructions)
  if(instructions=="HALT"):
      break
l = [x.upper() for x in l]
        # Converting all the instructions to upper case
for i in range(len(l)):
    x = l[i]
    if " " in x:
        s1 = ''.join(x)
        a, b = s1.split()
        if a in motOpCode:    # Checking if Mnemonics is present in MOT or not
            value = motOpCode.get(a)
            size = motSize.get(a)
            previous = size
            RA += current
            current = previous
            relativeAddress.append(RA)
            if b.isalpha() is True:
                machineCode.append(str(value))
            else:
                temp = list(b)
                for i in range(len(temp)):
                    if count == 2:
                        temp.insert(i, ',')
                        count = 0
                    else:
                        count = count + 1
                s = ''.join(temp)
```

```
            machineCode.append(str(value) + "," + s)
        else:
            print("Instruction is not in Op Code Table.")
            exit(0)      # EXIT if Mnemonics is not in MOT
    else:
        if x in motOpCode:
            value = motOpCode.get(x)
            size = motSize.get(x)
            previous = size
            RA += current
            current = previous
            relativeAddress.append(RA)
            machineCode.append(value)
        else:
            print("Instruction is not in Op Code Table.")
            exit(0)

print("Relative Address Instruction     OpCode")
for i in range(len(l)):
    print(
        "{}              {}         {}".format(relativeAddress[i], l[i], machineCode[i]))
```

OUTPUT :

```
Enter the starting address: 1000
Enter ALP :
MOV R
ADD R
SUB 30
STORE 1000
HALT
Relative Address Instruction     OpCode
1000                 MOV R          1
1001                 ADD R          8
1002                 SUB 30          9,30
1004                 STORE 1000          15,,10,00
1007                 HALT          20
```

CODE :

```
from sys import exit
motOpCode = {
    "MOV": 1,
    "A": 2,
    "S": 3,
    "M": 4,
    "D": 5,
    "AN": 6,
    "O": 7,
    "ADD": 8,
    "SUB": 9,
    "MUL": 10,
    "DIV": 11,
    "AND": 12,
    "OR": 13,
    "LOAD": 14,
    "STORE": 15,
    "DCR": 16,
    "INC": 17,
    "JMP": 18,
    "JNZ": 19,
    "HALT": 20}
motSize = {
    "MOV": 1,
    "A": 1,
    "S": 1,
    "M": 1,
    "D": 1,
    "AN": 1,
    "O": 1,
    "ADD": 1,
    "SUB": 2,
    "MUL": 2,
    "DIV": 2,
    "AND": 2,
    "OR ": 2,
    "LOAD": 3,
    "STORE": 3,
    "DCR": 1,
```

```
    "INC": 1,
    "JMP": 3,
    "JNZ": 3,
    "HALT": 1}
l = []
relativeAddress = []
machineCode = []
symbol = []
symbolValue = []
RA = int(input("Enter the starting address:"))
current = 0
count = 0
temp = []
print("Enter ALP:")
while True:
  instructions = input()
  l.append(instructions)
   if(instructions=="HALT"):
      break
l = [x.upper() for x in l]
for i in range(len(l)):
    x = l[i]
    if "NEXT:" in x:
        s1 = ''.join(x)
        a, b, c = s1.split()
        a = a[:4]
        l[i] = b + " " + c
        symbol.append(a)
        x = l[i]
        if b in motOpCode:
            value = motOpCode.get(b)
            size = motSize.get(b)
            if len(str(size)) == 1:
                temp = "000" + str(size)
            elif len(str(size)) == 2:
                temp = "00" + str(size)
            elif len(str(size)) == 3:
                temp = "0"+str(size)
        else:
            print("Instruction is not in Op Code.")
            exit(0)
```

```
    symbolValue.append(temp)
    previous = size
    RA += current
    current = previous
    relativeAddress.append(RA)
    if c.isalpha() is True:
        machineCode.append(str(value))
    else:
        temp = list(b)
        for i in range(len(temp)):
            if count == 2:
                temp.insert(i, ',')
                count = 0
            else:
                count = count + 1
        s = ''.join(temp)
        machineCode.append(str(value) + "," + s)
elif " " in x:
    s1 = ''.join(x)
    a, b = s1.split()
    if a in motOpCode:
        value = motOpCode.get(a)
        size = motSize.get(a)
        previous = size
        RA += current
        current = previous
        relativeAddress.append(RA)
        if b.isalpha() is True:
            machineCode.append(str(value))
        else:
            temp = list(b)
            for i in range(len(temp)):
                if count == 2:
                    temp.insert(i, ',')
                    count = 0
                else:
                    count = count + 1
            s = ''.join(temp)
            machineCode.append(str(value) + "," + s)
    else:
        print("Instruction is not in Op Code.")
```

```
            exit(0)
        else:
            if x in motOpCode:
                value = motOpCode.get(x)
                size = motSize.get(x)
                previous = size
                RA += current
                current = previous
                relativeAddress.append(RA)
                machineCode.append(value)
            else:
                print("Instruction is not in Op Code.")
                exit(0)
print("Symbol Table  :  \n")
print("\n Symbol          Value(Address)")
for i in range(len(symbol)):
    print(" {}              {}".format(symbol[i], symbolValue[i]))
print("\n Pass-1 machine code output without reference of the symbolic address : \n")
print("Relative Address Instruction     OpCode")
for i in range(len(l)):
    if "NEXT" in l[i]:
        print("{}                          {}              {}, - , - ".format(
            relativeAddress[i], l[i], machineCode[i]))
    else:
        print("{}                          {}              {} ".format(
            relativeAddress[i], l[i], machineCode[i]))
print("\n Pass-2 output: Machine code output \n ")
print("Relative Address Instruction     OpCode")
for i in range(len(l)):
    if "NEXT" in l[i]:
        for j in range(len(symbol)):
            if "NEXT" in symbol[j]:
                pos = j
                print("{}                          {}              {} , {}".format(
                    relativeAddress[i], l[i], machineCode[i], symbolValue[pos]))
else:
        print("{}                          {}              {} ".format(
            relativeAddress[i], l[i], machineCode[i]))
```

OUTPUT :

```
Enter the starting address:1000
Enter ALP:
MOV R
Next: ADD R
DCR R
JNZ Next
STORE 1000
HALT
Symbol Table  :


 Symbol           Value(Address)
 NEXT              0001

 Pass-1 machine code output without reference of the symbolic address :

Relative Address Instruction     OpCode
1000                                 MOV R                1
1001                                 ADD R                8
1002                                 DCR R                16
1003                                 JNZ NEXT                19, - , -
1006                                 STORE 1000                15,10,00
1009                                 HALT                20

 Pass-2 output: Machine code output

Relative Address Instruction     OpCode
1000                                 MOV R                1
1001                                 ADD R                8
1002                                 DCR R                16
1003                                 JNZ NEXT                19 , 0001
1006                                 STORE 1000                15,10,00
1009                                 HALT                20
```