```
def generate_symbol_table(assembly_code):
    symbol_table = {}

    # First Pass: Collect labels and their addresses
    current_address = None
    for line in assembly_code:
        parts = line.split()
        if parts[0] == "START":
            current_address = int(parts[1])
        elif parts[0] != "END":
            if(len(parts)==2):
              if(parts[0]=="READ"):
                symbol_table[parts[1]]="Not assigned"
            elif(len(parts)==4):
              symbol_table[parts[0]]=current_address
            elif(len(parts)==3 and parts[1]=="DS"):
              symbol_table[parts[0]]=current_address
              current_address+=int(parts[2])-1
            elif(len(parts)==3 and parts[1]=="DC"):
              symbol_table[parts[0]]=current_address
            current_address+=1


    # Second Pass: Generate Symbol Table
    print("Symbol Table:")
    print("Label\t|\tAddress")
    print("----------------------")
    for label, address in symbol_table.items():
        if label=="READ":
          continue
        print(f"{label}\t|\t{address}")

# Assembly code
assembly_code = [
    "START 100",
    "READ A",
    "READ B",
    "LOOP MOVER AREG, A",
    "MOVER BREG, B",
    "COMP BREG, ='2'",
    "BC GT, LOOP",
    "BACK SUB AREG, B",
    "COMP AREG, ='5'",
    "BC LT, BACK",
    "STOP",
    "A DS 1",
    "print suyash",
    "B DS 1",
    "END"
]

# Generate symbol table
generate_symbol_table(assembly_code)
```

```
Symbol Table:
Label   |       Address
----------------------
A       |       110
B       |       112
LOOP    |       102
BACK    |       106
```

```
def generate_symbol_table(assembly_code):
    symbol_table = {}

    # First Pass: Collect labels and their addresses
    current_address = None
    for line in assembly_code:
        parts = line.split()
        if parts[0] == "START":
            current_address = int(parts[1])
        elif parts[0] != "END":
            if(len(parts)==2):
              if(parts[0]=="READ"):
                symbol_table[parts[1]]="Not assigned"
            elif(len(parts)==4):
              symbol_table[parts[0]]=current_address
            elif(len(parts)==3 and parts[1]=="DS"):
              symbol_table[parts[0]]=current_address
              current_address+=int(parts[2])-1
            elif(len(parts)==3 and parts[1]=="DC"):
              symbol_table[parts[0]]=current_address
            current_address+=1


    # Second Pass: Generate Symbol Table
    print("Symbol Table:")
    print("Label\t|\tAddress")
    print("----------------------")
    for label, address in symbol_table.items():
        if label=="READ":
          continue
        print(f"{label}\t|\t{address}")

# Assembly code
assembly_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"
]

# Generate symbol table
generate_symbol_table(assembly_code)
```

```
Symbol Table:
Label   |       Address
----------------------
A       |       113
B       |       114
LOOP    |       105
NEXT    |       109
```

```
def generate_symbol_table(assembly_code):
    symbol_table = {}

    # First Pass: Collect labels and their addresses
    current_address = None
    for line in assembly_code:
        parts = line.split()
        if parts[0] == "START":
            current_address = int(parts[1])
        elif parts[0] != "END":
            if(len(parts)==2):
              if(parts[0]=="READ"):
                symbol_table[parts[1]]="Not assigned"
            elif(len(parts)==4):
              symbol_table[parts[0]]=current_address
              #current_address+=1
            elif(len(parts)==3 and parts[1]=="DS"):
              symbol_table[parts[0]]=current_address
              current_address+=int(parts[2])-1
            elif(len(parts)==3 and parts[1]=="DC"):
              symbol_table[parts[0]]=current_address
            current_address+=1


    # Second Pass: Generate Symbol Table
    print("Symbol Table:")
    print("Label\t|\tAddress")
    print("----------------------")
    for label, address in symbol_table.items():
        if label=="READ":
          continue
        print(f"{label}\t|\t{address}")

# Assembly code
assembly_code = [
    "START 150",
    "READ D",
    "READ E",
    "LOOP MOVER AREG, D",
    "MOVER BREG, E",
    "COMP BREG, ='20'",
    "BC GT, LOOP",
    "BACK SUB AREG, E",
    "COMP AREG, ='50'",
    "BC LT, BACK",
    "STOP",
    "D DS 1",
    "E DS 1",
    "END"
]

# Generate symbol table
generate_symbol_table(assembly_code)
```

```
Symbol Table:
Label   |       Address
----------------------
D       |       160
```