In [1]:
```
from qiskit import*
from qiskit.visualization import plot_histogram
from qiskit.visualization import plot_bloch_multivector
from qiskit import QuantumCircuit, Aer, transpile, assemble, execute
import numpy as np
from qiskit.visualization import visualize_transition
from qiskit.visualization import plot_bloch_multivector
```

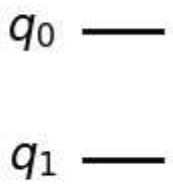
In [2]:
```
#Creating a constant Oracle input has no effect on the output
#Create a quantum circuit with 2 qubits. one as input and other as output
constant_oracle = QuantumCircuit(2)
```

In [3]:
```
# get a random number from 0 or1
output = np.random.randint(2)
```

In [15]:
```
#what ever get in input , its having no effect
# the output will be the random value 0 or 1
if output == 1:
    constant_oracle.x(1)
```

In [16]:
```
#draw the circuit
constant_oracle.draw('mpl')
```

Out[16]:

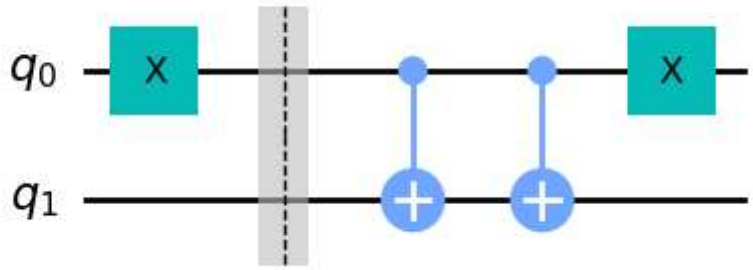


In [17]:
```
#creating a balanced oracle
# perform CNOTs with first qubit input as a control and the second qubit outpu
balanced_oracle = QuantumCircuit(2)
# place X-gate for input qubit
balanced_oracle.x(0)
# use a barrier as divider
balanced_oracle.barrier()
```

Out[17]: <qiskit.circuit.instructionset.InstructionSet at 0x1d944d50370>

In [18]:
```
#Place controlled-Not gates
balanced_oracle.cx(0,1)
# using barrier as a divider and avoid cancelling gates by the transpiler
balanced_oracle.cx(0,1)
#place X-gates
balanced_oracle.x(0)
#show oracle
balanced_oracle.draw('mpl')
```

Out[18]:



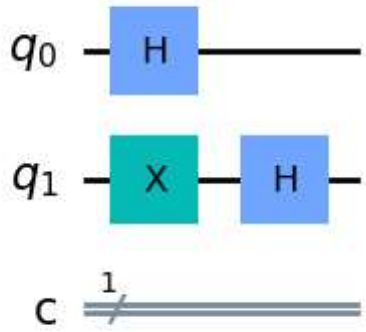
In [19]:
```
#Initialize the input qubits in the state |+>
# and the out qubit in the state |->
dj_circuit = QuantumCircuit(2,1)
```

In [20]:
```
# Apply H-gate on q0
dj_circuit.h(0)
```

Out[20]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x1d944d508b0>
```

In [21]:
```
# put the qubit in state |->
# qubit q1 initialize it with X gate first in order to flip its state at the s
dj_circuit.x(1)
dj_circuit.h(1)
dj_circuit.draw('mpl')
```

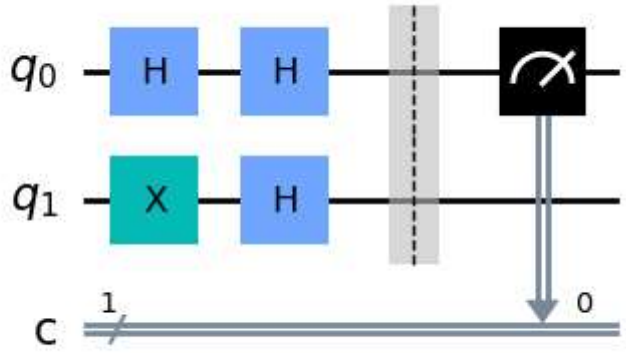
Out[21]:



In [22]:
```
constant_oracle = QuantumCircuit(2)
#balanced_oracle = QuantumCircuit(2)
# define the oracle function to use
oracle_fn = constant_oracle
#oracle_fn = balanced_oracle
```

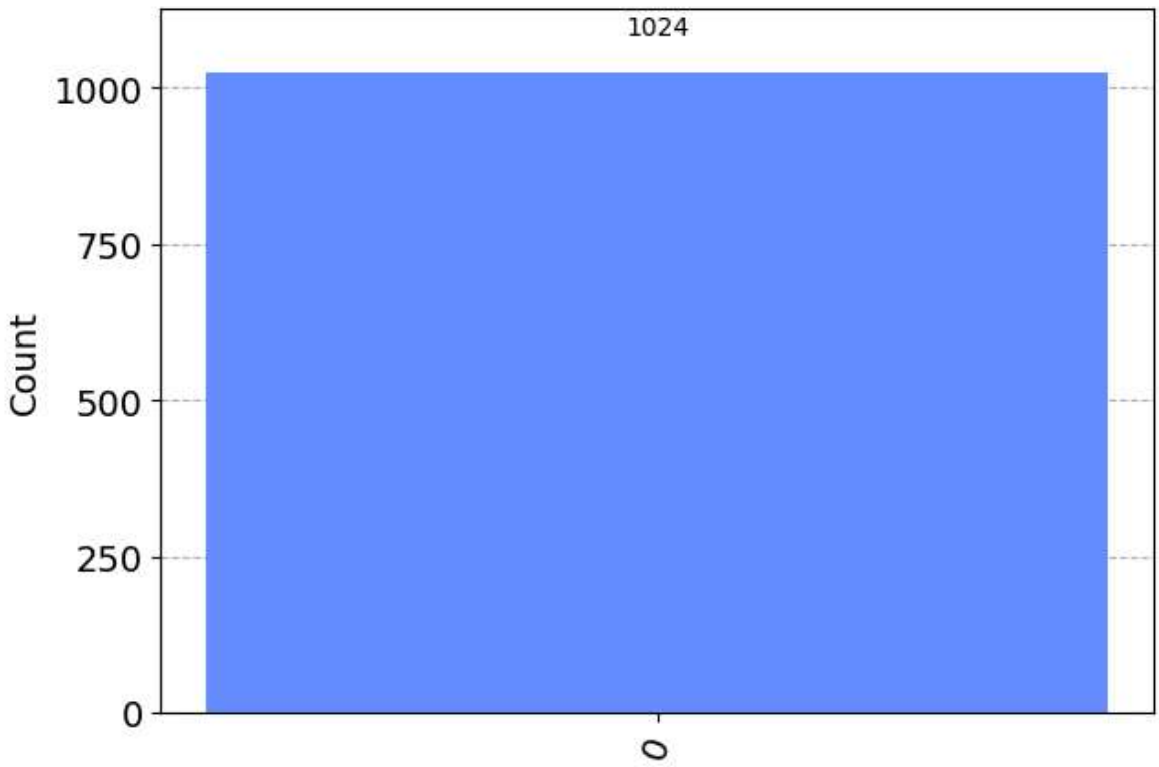
In [23]:
```
# perform H-gates on qubit and measure input register
dj_circuit.h(0)
dj_circuit.barrier()
#measure
dj_circuit.measure(0,0)
#Display circuit
dj_circuit.draw('mpl')
```

Out[23]:



In [24]:
```
#check the output
# use local simulator
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(dj_circuit, backend=backend, shots=shots).result()
answer = results.get_counts()
plot_histogram(answer)
```

Out[24]:



In [ ]:

In [ ]: