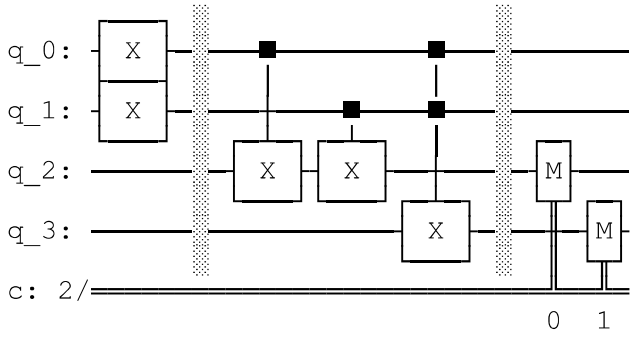
In [2]:
```
from qiskit import QuantumCircuit, assemble, Aer
from qiskit.visualization import plot_histogram
```

In [3]:
```
#ADD 1 AND 1

qc_ha = QuantumCircuit(4,2)
qc_ha.x(0)
qc_ha.x(1)
qc_ha.barrier()
qc_ha.cx(0,2)
qc_ha.cx(1,2)
qc_ha.ccx(0,1,3)
qc_ha.barrier()
qc_ha.measure(2,0)
qc_ha.measure(3,1)
qc_ha.draw()
```

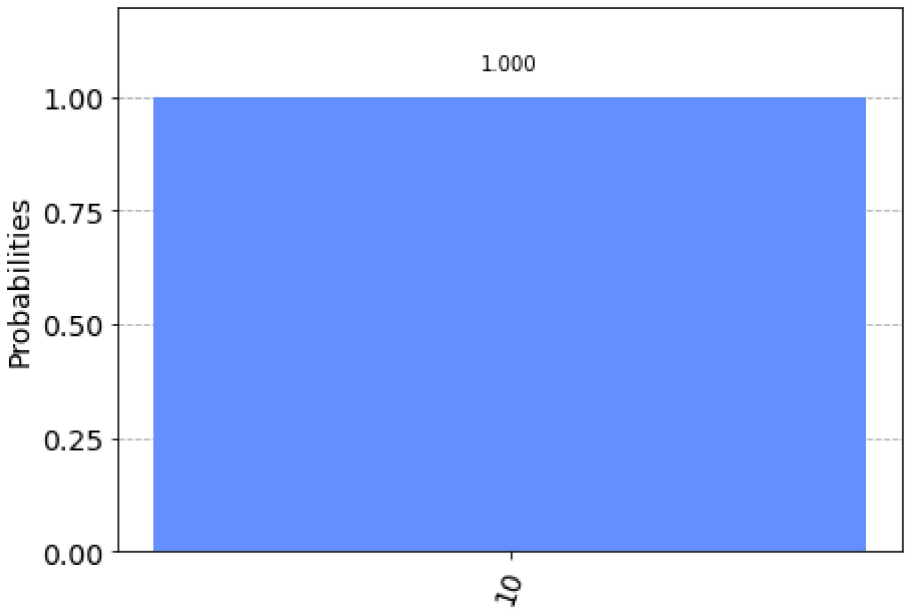
Out[3]:



In [4]:
```
sim = Aer.get_backend('qasm_simulator')
qobj = assemble(qc_ha)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)

#OUTPUT OF 1+1 IS 10
```

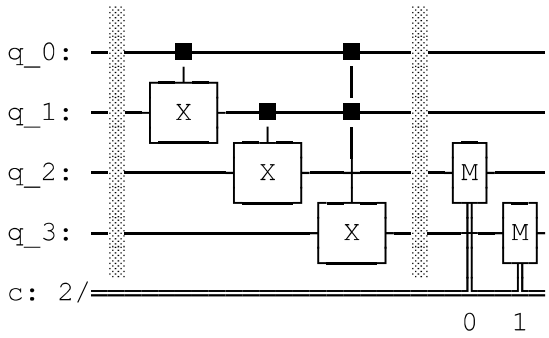
Out[4]:



In [7]:
```
#ADD 0 AND 0

a = QuantumCircuit(4,2)
a.barrier()
a.cx(0,1)
a.cx(1,2)
a.ccx(0,1,3)
a.barrier()
a.measure(2,0)
a.measure(3,1)
a.draw()
```

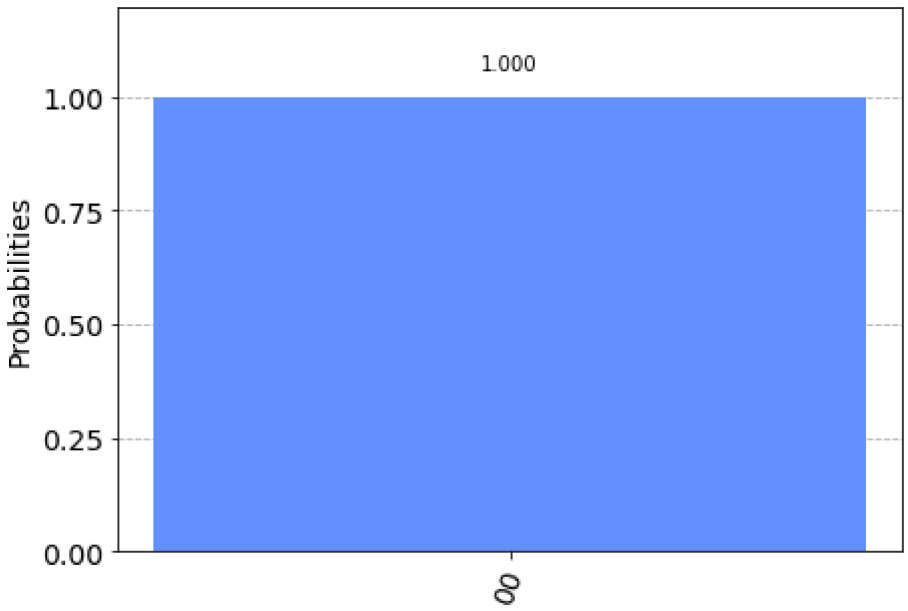
Out[7]:



In [8]:
```
sim = Aer.get_backend('qasm_simulator')
qobj = assemble(a)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)

#OUTPUT OF 0+0 IS 00
```

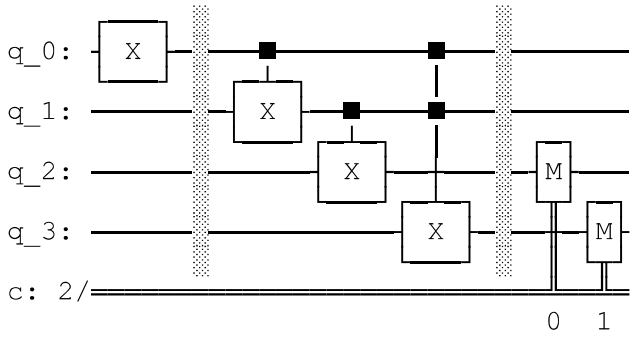
Out[8]:



In [9]:
```
#ADD 1 AND 0

b = QuantumCircuit(4,2)
b.x(0)
b.barrier()
b.cx(0,1)
b.cx(1,2)
b.ccx(0,1,3)
b.barrier()
b.measure(2,0)
b.measure(3,1)
b.draw()
```

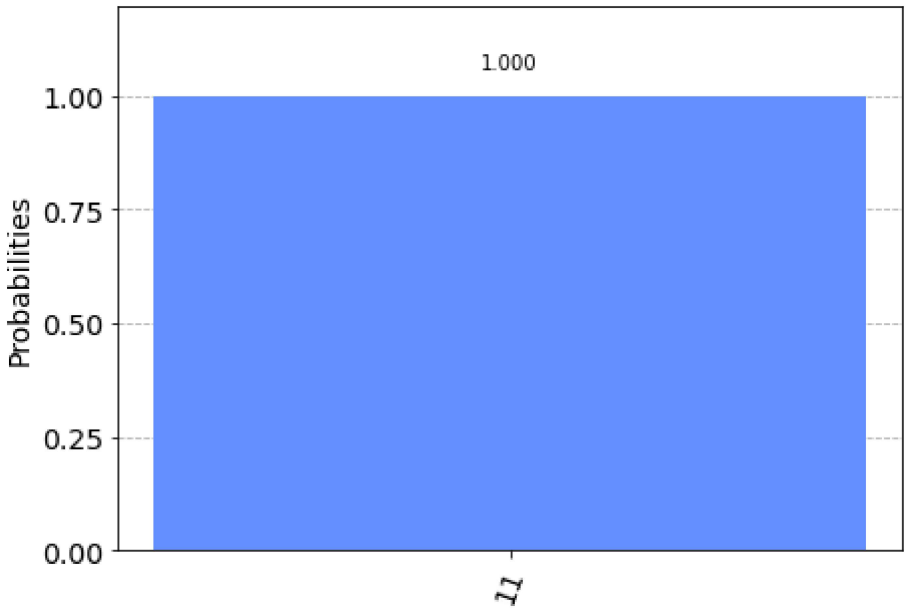
Out[9]:



In [10]:
```
sim = Aer.get_backend('qasm_simulator')
qobj = assemble(b)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)

#OUTPUT OF 1+0 IS 11
```

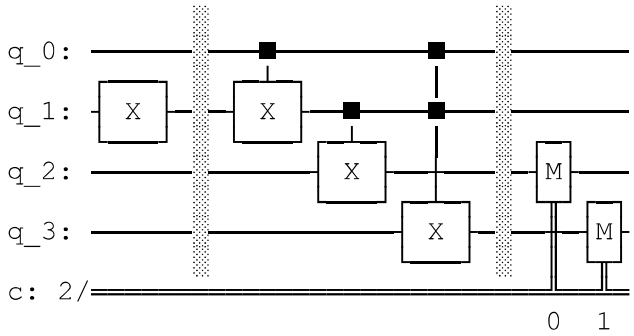
Out[10]:



In [11]:
```
#ADD 0 AND 1

c = QuantumCircuit(4,2)
c.x(1)
c.barrier()
c.cx(0,1)
c.cx(1,2)
c.ccx(0,1,3)
c.barrier()
c.measure(2,0)
c.measure(3,1)
c.draw()
```

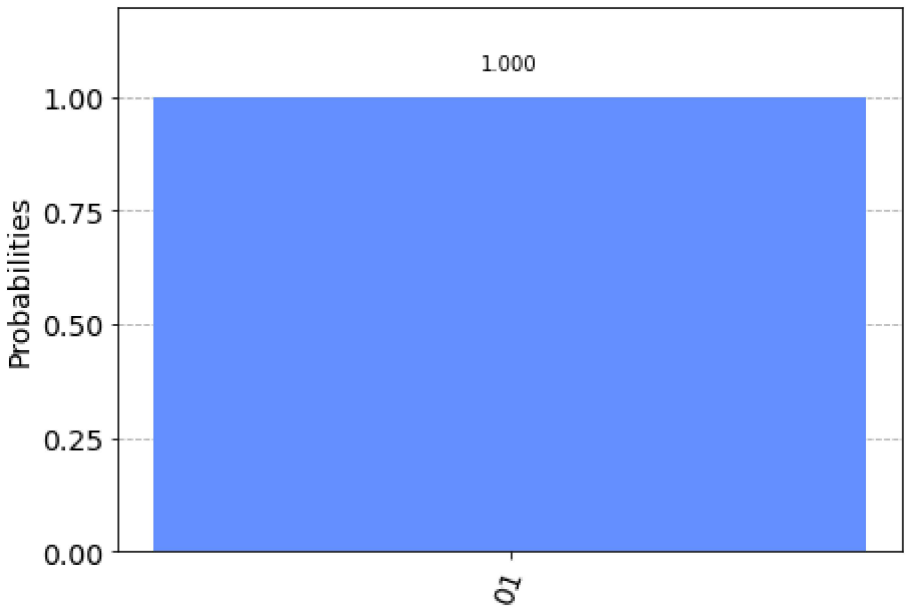
Out[11]:



In [12]:
```
sim = Aer.get_backend('qasm_simulator')
qobj = assemble(c)
result = sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)

#OUTPUT OF 0+1 IS 01
```

Out[12]:



In [ ]: