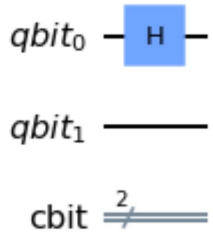
In [1]:
```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit, transpi
from qiskit.visualization import plot_histogram, plot_bloch_vector
from qiskit.quantum_info import Statevector
from qiskit import Aer
from qiskit import BasicAer
```

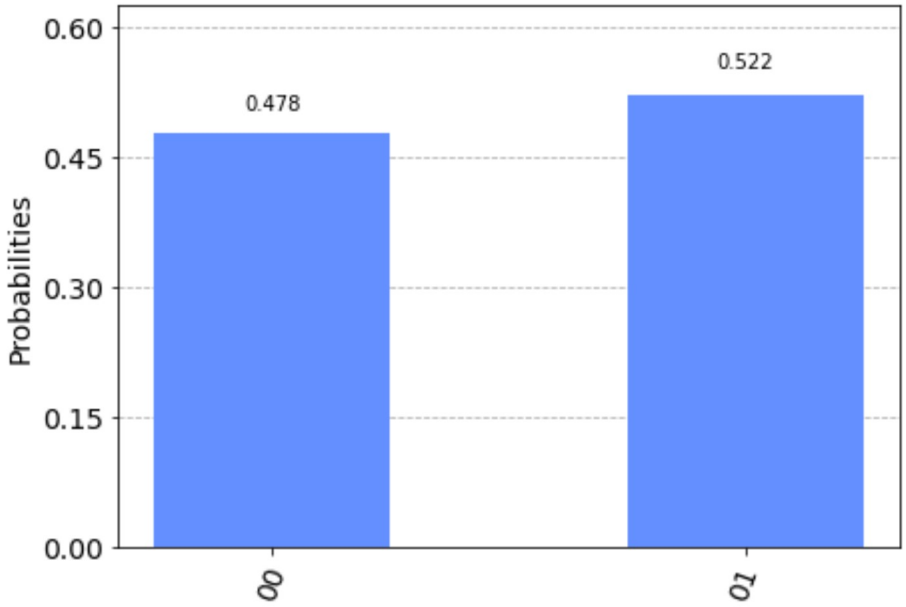
Bell state 1

In [2]:
```
qr = QuantumRegister(size = 2 , name = "qbit")
cr = ClassicalRegister(size = 2, name = "cbit")
q_circ = QuantumCircuit(qr, cr)
q_circ.h(0)
display(q_circ.draw(output = 'mpl'))
```



In [3]:
```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[3]:



In [4]:

```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

Out[4]:

$$\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|01\rangle$$

In [5]:

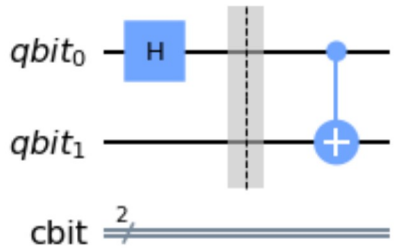
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[5]:

```
array([[ 0.707+0.j,  0.707-0.j,  0.   +0.j,  0.   +0.j],
       [ 0.707+0.j, -0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j,  0.707-0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j, -0.707+0.j]])
```

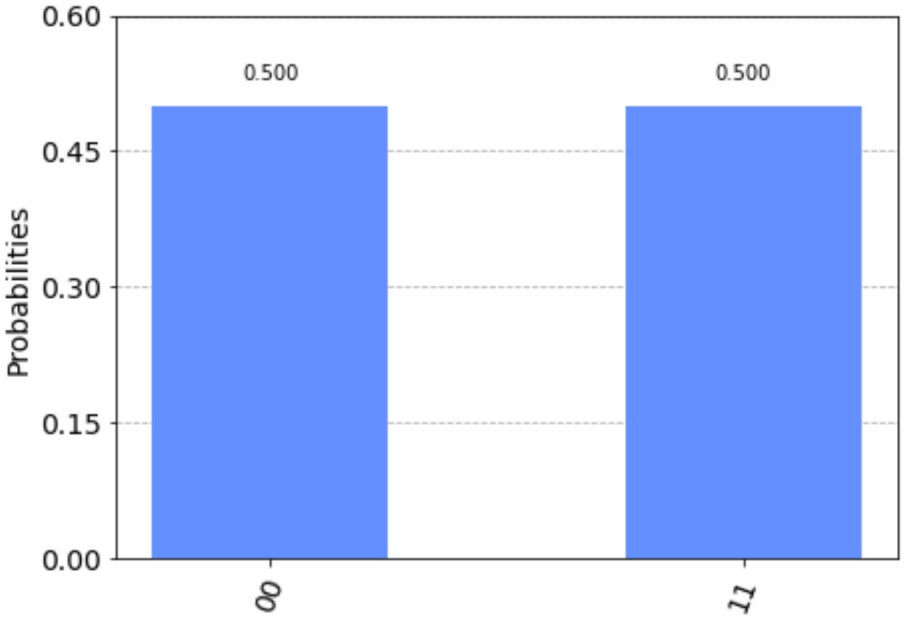
In [6]:

```
q_circ.barrier()
q_circ.cx(0,1)
display(q_circ.draw(output = 'mpl'))
```



In [7]:
```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[7]:



In [8]:
```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

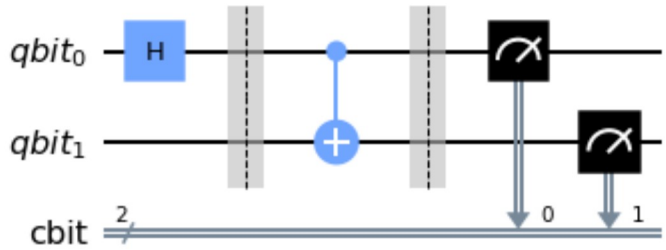
Out[8]:

$$\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$$

In [9]:
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[9]:
```
array([[ 0.707+0.j,  0.707-0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j, -0.707+0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j,  0.707-0.j],
       [ 0.707+0.j, -0.707+0.j,  0.   +0.j,  0.   +0.j]])
```

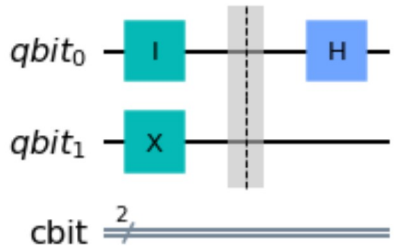
In [10]:
```
display(new_circ.draw('mpl'))
```



Bell state 2

In [11]:

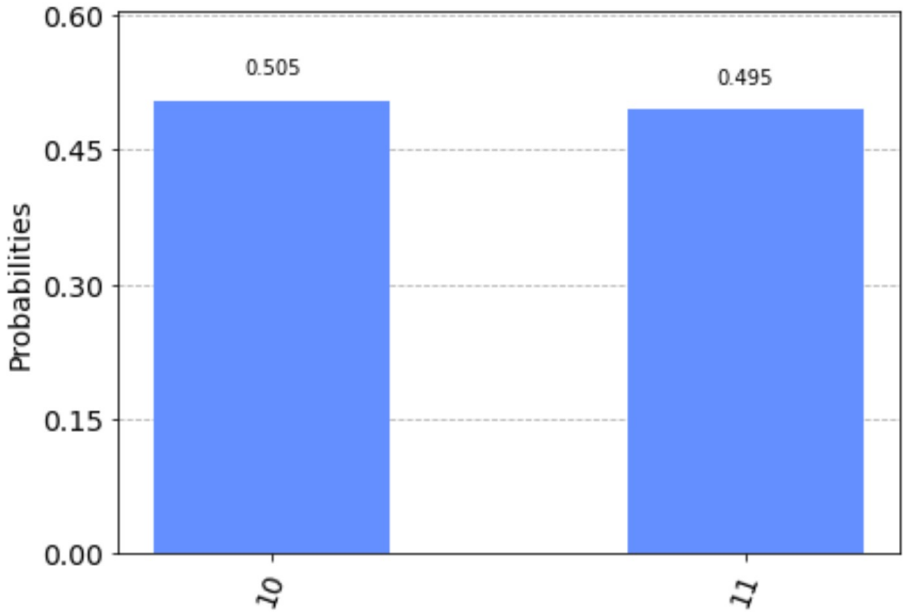
```
qr = QuantumRegister(size = 2 , name = "qbit")
cr = ClassicalRegister(size = 2, name = "cbit")
q_circ = QuantumCircuit(qr, cr)
q_circ.x(1)
q_circ.i(0)
q_circ.barrier()
q_circ.h(0)
display(q_circ.draw(output = 'mpl'))
```



In [12]:

```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[12]:



In [13]:

```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

Out[13]:

$$\frac{\sqrt{2}}{2}|10\rangle + \frac{\sqrt{2}}{2}|11\rangle$$

In [14]:

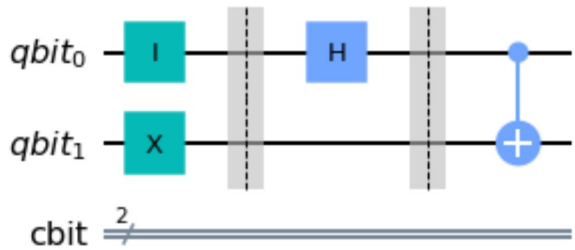
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[14]:

```
array([[ 0.   +0.j,  0.   +0.j,  0.707+0.j,  0.707-0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j, -0.707+0.j],
       [ 0.707+0.j,  0.707-0.j,  0.   +0.j,  0.   +0.j],
       [ 0.707+0.j, -0.707+0.j,  0.   +0.j,  0.   +0.j]])
```

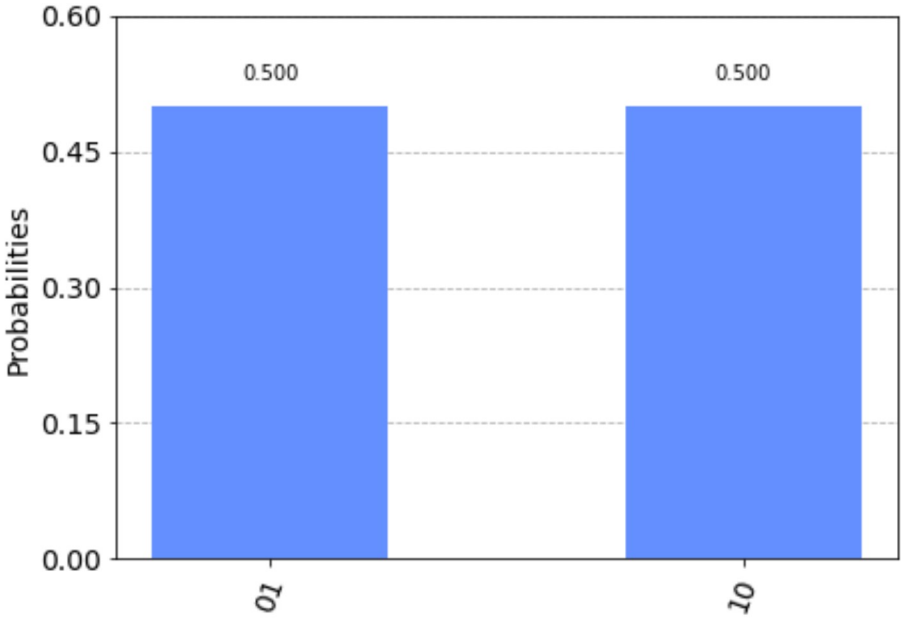
In [15]:

```
q_circ.barrier()
q_circ.cx(0,1)
display(q_circ.draw(output = 'mpl'))
```



In [16]:
```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[16]:



In [17]:
```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

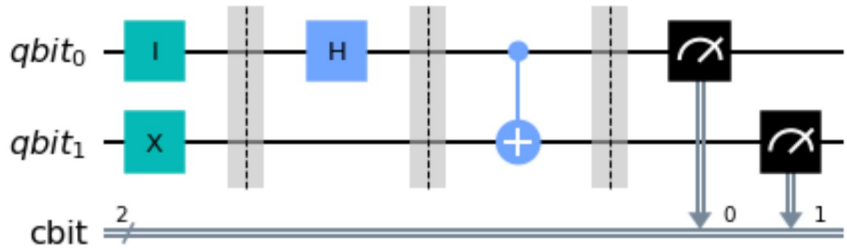
Out[17]:

$$\frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle$$

In [18]:
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[18]:
```
array([[ 0.   +0.j,  0.   +0.j,  0.707+0.j,  0.707-0.j],
       [ 0.707+0.j, -0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.707+0.j,  0.707-0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707+0.j, -0.707+0.j]])
```

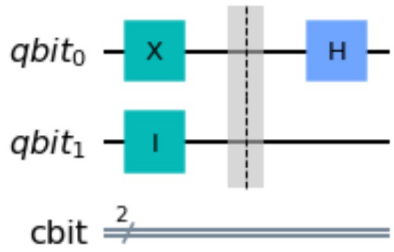
In [19]:
```
display(new_circ.draw('mpl'))
```



Bell state 3

In [20]:

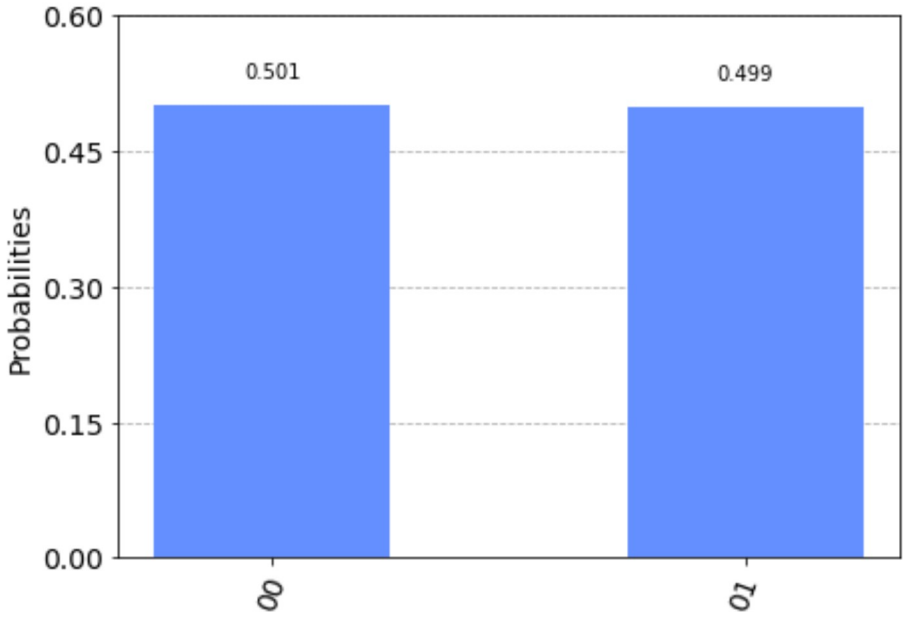
```
qr = QuantumRegister(size = 2 , name = "qbit")
cr = ClassicalRegister(size = 2, name = "cbit")
q_circ = QuantumCircuit(qr, cr)
q_circ.x(0)
q_circ.i(1)
q_circ.barrier()
q_circ.h(0)
display(q_circ.draw(output = 'mpl'))
```



In [21]:

```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[21]:



In [22]:

```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

Out[22]:

$$\frac{\sqrt{2}}{2}|00\rangle - \frac{\sqrt{2}}{2}|01\rangle$$

In [23]:

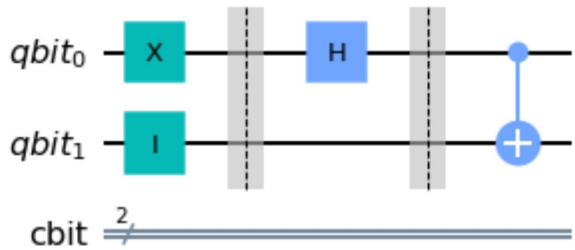
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[23]:

```
array([[ 0.707-0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [-0.707+0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707-0.j,  0.707+0.j],
       [ 0.   +0.j,  0.   +0.j, -0.707+0.j,  0.707+0.j]])
```

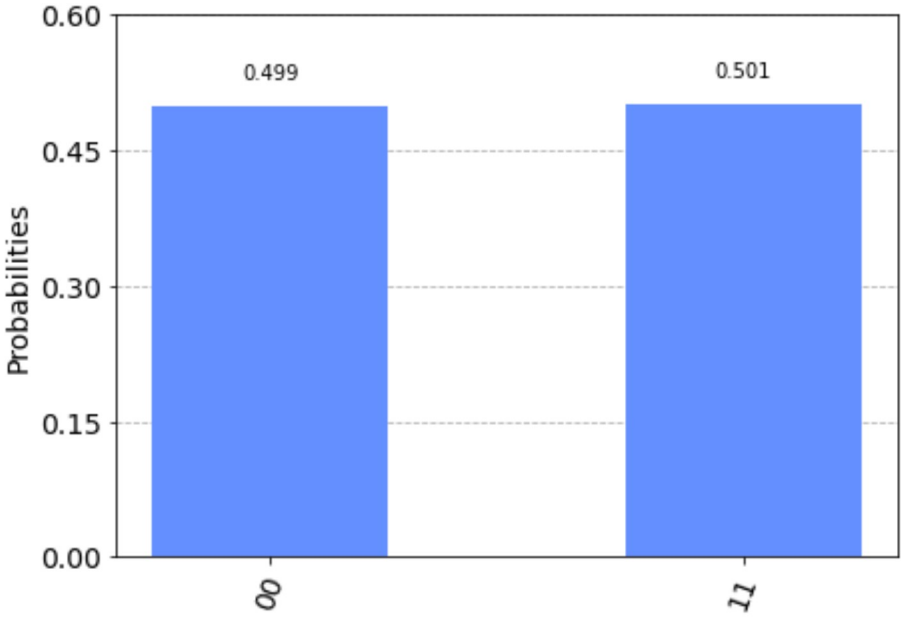
In [24]:

```
q_circ.barrier()
q_circ.cx(0,1)
display(q_circ.draw(output = 'mpl'))
```



In [25]:
```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[25]:



In [26]:
```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

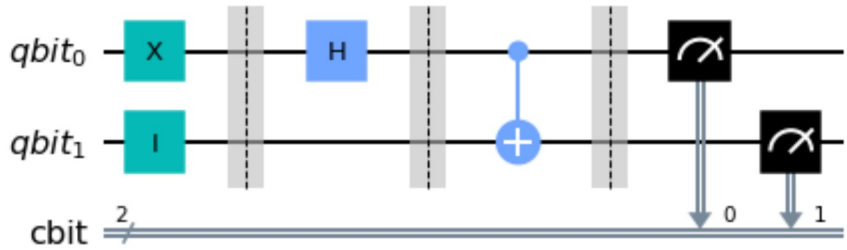
Out[26]:

$$\frac{\sqrt{2}}{2}|00\rangle - \frac{\sqrt{2}}{2}|11\rangle$$

In [27]:
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[27]:
```
array([[ 0.707-0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j, -0.707+0.j,  0.707+0.j],
       [ 0.   +0.j,  0.   +0.j,  0.707-0.j,  0.707+0.j],
       [-0.707+0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j]])
```

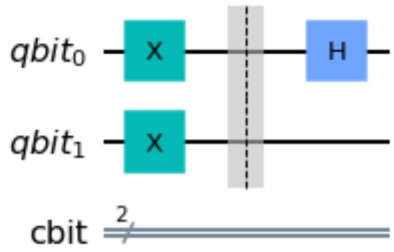
In [28]:
```
display(new_circ.draw('mpl'))
```



Bell state 4

In [29]:

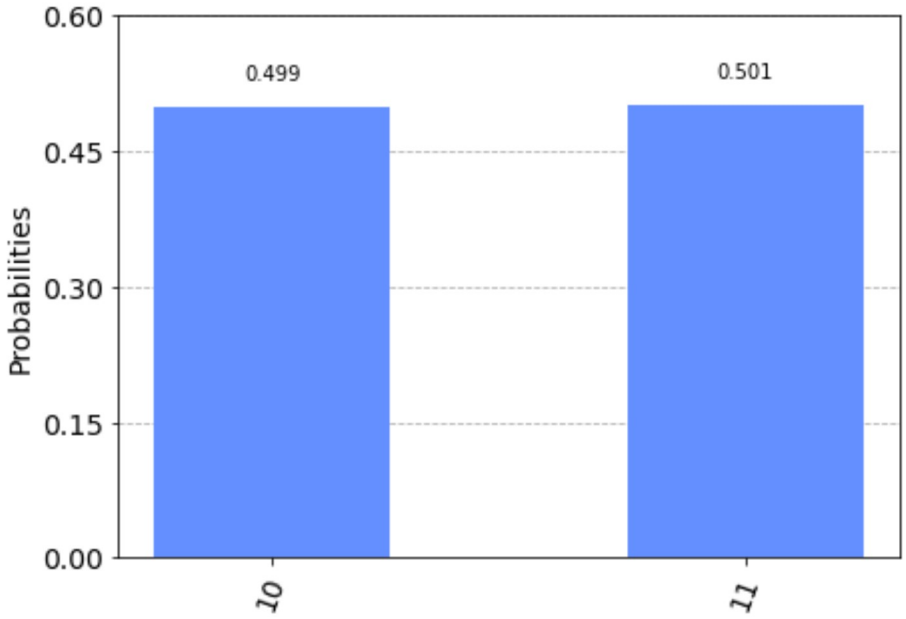
```
qr = QuantumRegister(size = 2 , name = "qbit")
cr = ClassicalRegister(size = 2, name = "cbit")
q_circ = QuantumCircuit(qr, cr)
q_circ.x(0)
q_circ.x(1)
q_circ.barrier()
q_circ.h(0)
display(q_circ.draw(output = 'mpl'))
```



In [30]:

```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[30]:



In [31]:

```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

Out[31]:

$$\frac{\sqrt{2}}{2}|10\rangle - \frac{\sqrt{2}}{2}|11\rangle$$

In [32]:

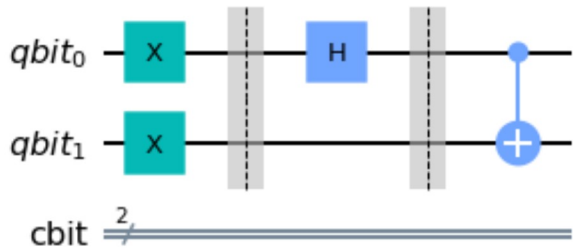
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[32]:

```
array([[ 0.   +0.j,  0.   +0.j,  0.707-0.j,  0.707+0.j],
       [ 0.   +0.j,  0.   +0.j, -0.707+0.j,  0.707+0.j],
       [ 0.707-0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [-0.707+0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j]])
```

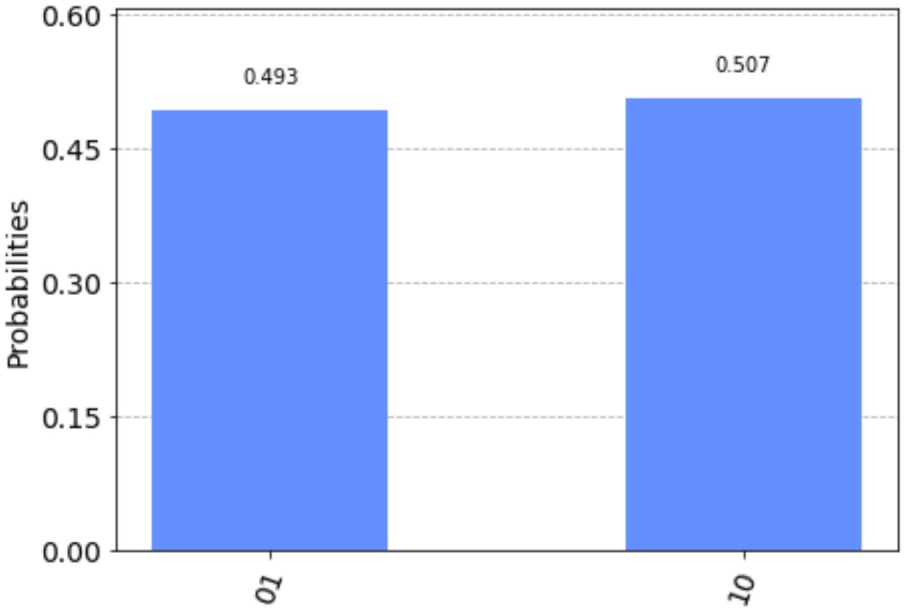
In [33]:

```
q_circ.barrier()
q_circ.cx(0,1)
display(q_circ.draw(output = 'mpl'))
```



In [34]:
```
meas = QuantumCircuit(qr)
meas.measure_all()
new_circ = q_circ.compose(meas)
sim_backend = Aer.get_backend('aer_simulator')
compiled_circ = transpile(new_circ, sim_backend)
#print(compiled_circ.draw())
result = sim_backend.run(compiled_circ, shots = 1024).result()
counts = result.get_counts(q_circ)
plot_histogram(counts)
```

Out[34]:



In [35]:
```
ket = Statevector(q_circ)
ket.draw(output = 'latex')
```

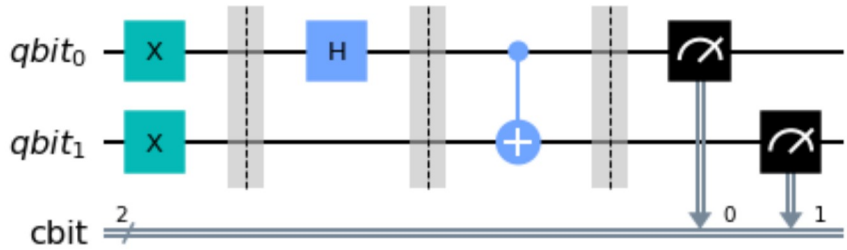
Out[35]:

$$-\frac{\sqrt{2}}{2}|01\rangle + \frac{\sqrt{2}}{2}|10\rangle$$

In [36]:
```
backend = BasicAer.get_backend('unitary_simulator')
job = backend.run(transpile(q_circ, backend))
job.result().get_unitary(q_circ, decimals=3)
```

Out[36]:
```
array([[ 0.   +0.j,  0.   +0.j,  0.707-0.j,  0.707+0.j],
       [-0.707+0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.707-0.j,  0.707+0.j,  0.   +0.j,  0.   +0.j],
       [ 0.   +0.j,  0.   +0.j, -0.707+0.j,  0.707+0.j]])
```

In [37]:
```
display(new_circ.draw('mpl'))
```



In [ ]:

In [ ]:

In [ ]: