In [22]:
```
from qiskit import IBMQ
```

In [11]:
```
import qiskit as qk
from qiskit import QuantumCircuit, BasicAer, execute
from qiskit.tools.visualization import plot_histogram
from math import pi
from qiskit.tools.visualization import plot_histogram
from qiskit.tools.monitor import job_monitor
from qiskit.providers.ibmq import least_busy
```

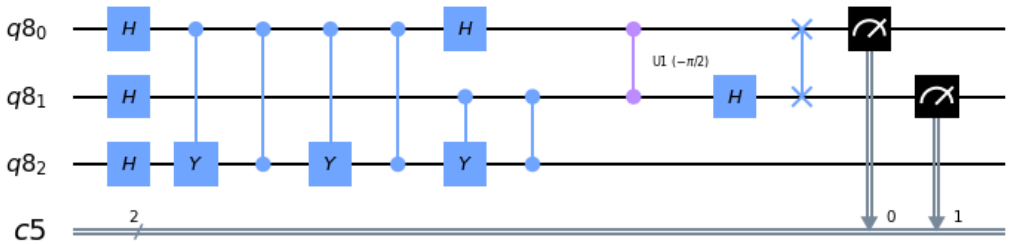
In [13]:
```
provider = IBMQ.load_account()
backend_sim = provider.get_backend('ibmq_qasm_simulator')
backend_lon = provider.get_backend('ibmq_london')#5 qubits here
```

In [12]:
```
def get_lb(min_qibuts):
    large_enough_devices = provider.backends(filters = lambda x: x.configuratio
n().n_qubits >= min_qibuts and not x.configuration().simulator)
    lb = least_busy(large_enough_devices)
    print("The best backend is"+ lb.name())
    return lb
```

In [32]:
```
def CU(qc, c_reg, t_reg):
    qc.cy(c_reg, t_reg)
    qc.cz(c_reg, t_reg)
    return qc

def QFT_d(qc, qr):
    qc.h(qr[0])
    qc.cu1(-pi/2.0,qr[0],qr[1])
    qc.h(qr[1])
    qc.swap(qr[0],qr[1])
    return qc
```

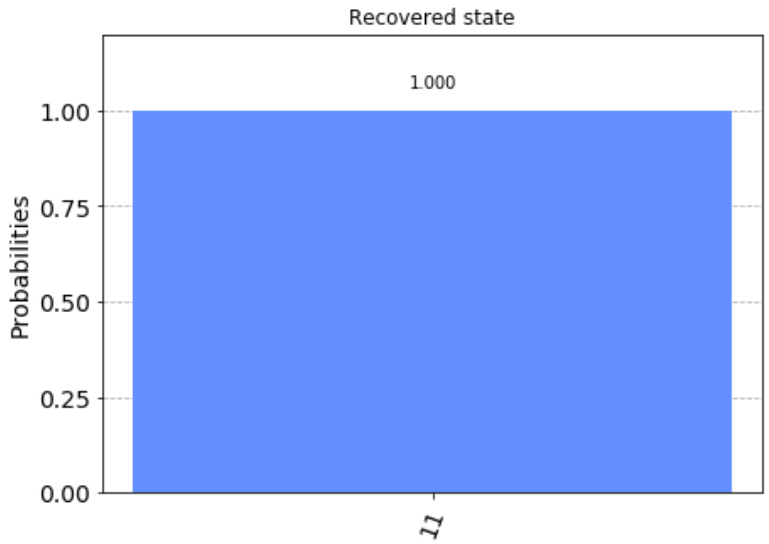
In [33]:
```
qr = qk.QuantumRegister(3)
cr = qk.ClassicalRegister(2)
qc = qk.QuantumCircuit(qr,cr)
qc.h(qr)
qc = CU(qc, qr[0], qr[2])
qc = CU(qc, qr[0], qr[2])
qc = CU(qc, qr[1], qr[2])
qc = QFT_d(qc,qr)

qc.measure(qr[0], cr[0])
qc.measure(qr[1], cr[1])
qc.draw(output='mpl')
```

Out[33]:



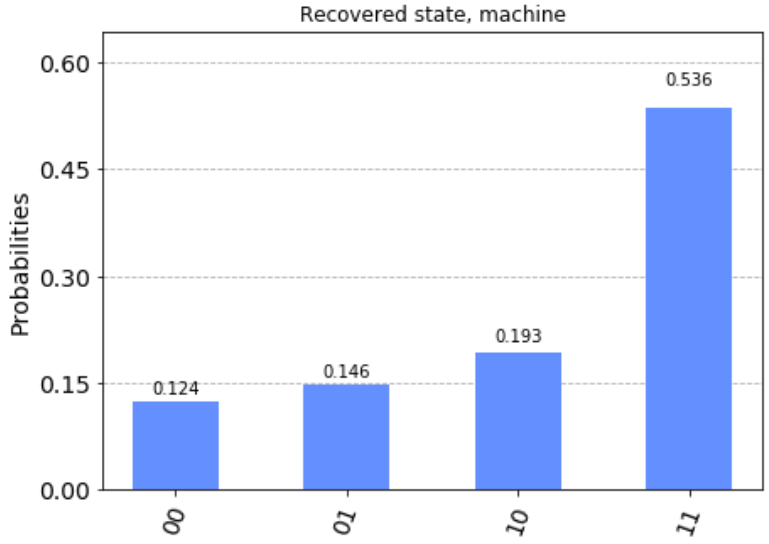
In [34]:
```
result = execute(qc, backend_sim, shots = 1024).result()
counts = result.get_counts(qc)
plot_histogram(counts, title='Recovered state')
```

Out[34]:



In [26]:
```
result = execute(qc, backend_lon, shots = 1024).result()
counts = result.get_counts(qc)
plot_histogram(counts, title='Recovered state, machine')
```

Out[26]:



In [ ]: