ADITHYA DUTT KAMBHAMPATI

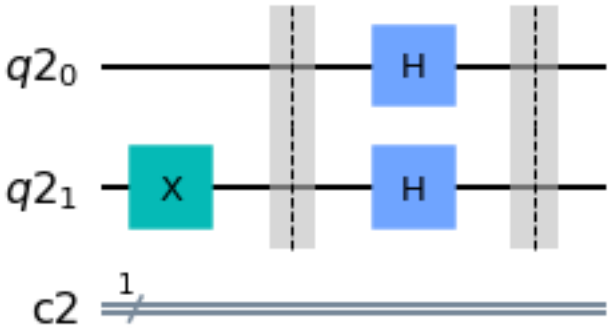
RA2011003011072

QUANTUM COMPUTATION CT 3 LAB ASSIGNMENT

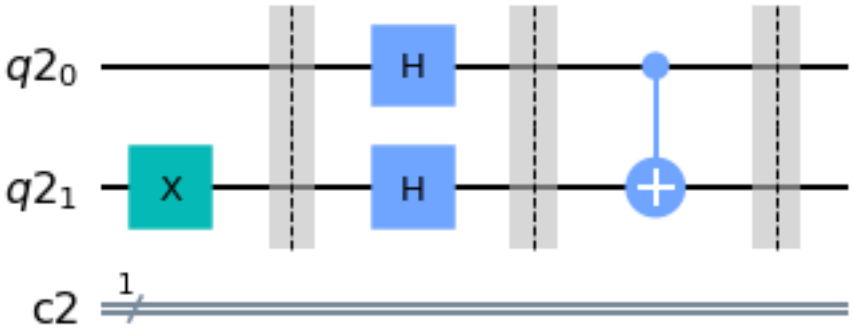
1. **Demonstrate Deutsch Algorithm for oracular function $f(0) = 1$ & $f(1) = 0$.**

   Code:

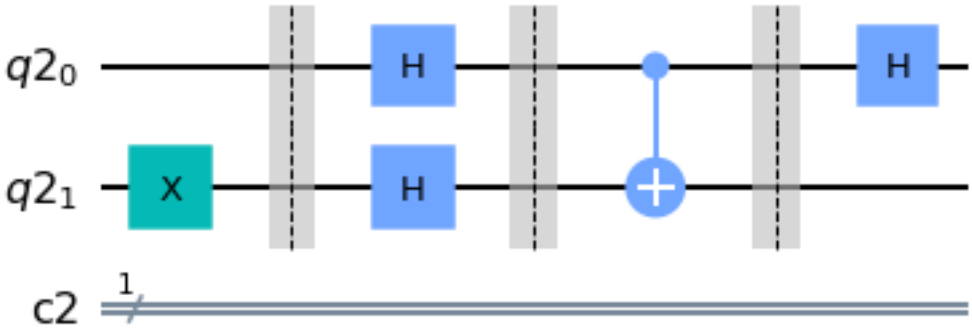
```
import numpy as np

# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ,
QuantumRegister, ClassicalRegister, execute, BasicAer
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit.providers.aer import QasmSimulator

# Loading your IBM Quantum account(s)
provider = IBMQ.load_account()

backend = BasicAer.get_backend('qasm_simulator')
shots=1024

qreg1 = QuantumRegister(2) # The quantum register of the qubits, in this
case 2 qubits
register1 = ClassicalRegister(1)

qc = QuantumCircuit(qreg1, register1)
qc.x(1)
qc.barrier()
qc.h([0,1])
qc.barrier()
qc.draw()
```



qc.cx(0,1)
qc.barrier()
qc.draw()

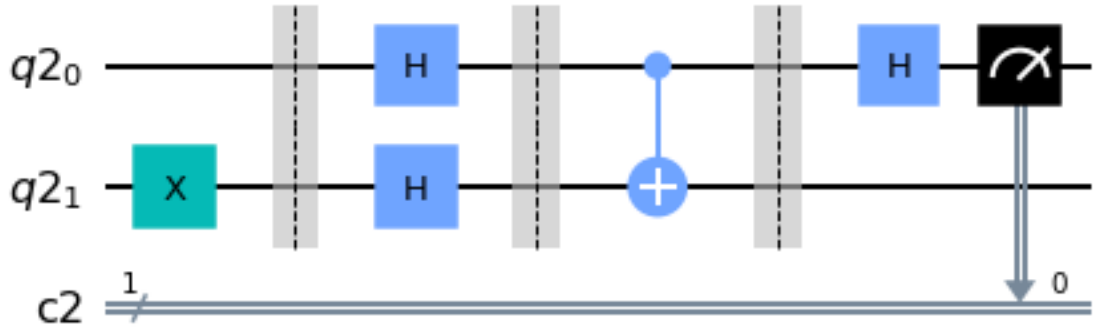


qc.h(0)
qc.draw()



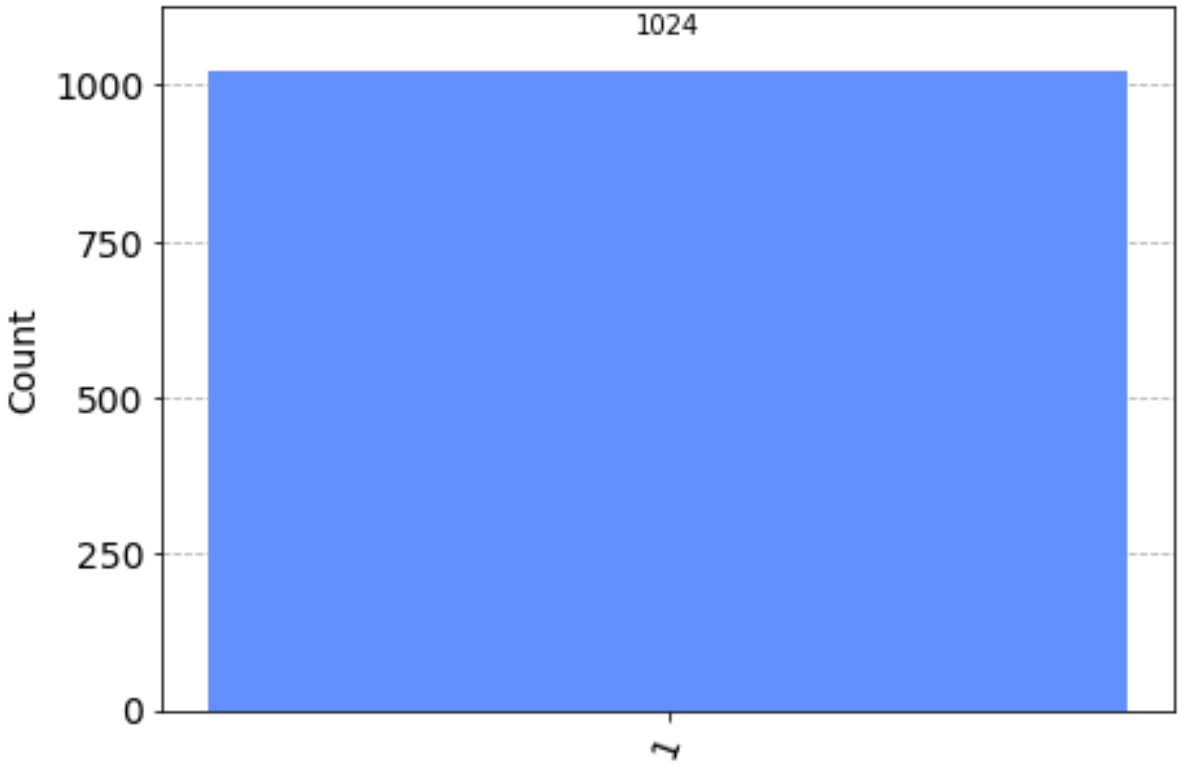
qc.measure(qreg1[0],register1)

qc.draw()



results = execute(qc, backend=backend, shots=shots).result()
answer = results.get_counts()

plot_histogram(answer)

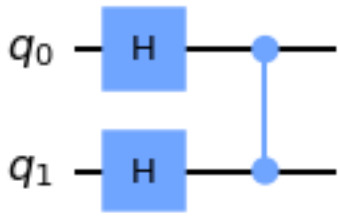


3. **Demonstrate Grover's Algorithm for $w = 01$ as the two-bit special string. Create the entire circuit and showcase how the circuit reveals the secret string hidden in the oracle.**

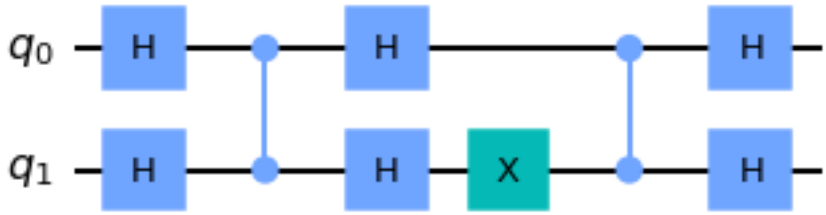
Code:

```
import numpy as np

# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from qiskit.providers.aer import QasmSimulator

# Loading your IBM Quantum account(s)
provider = IBMQ.load_account()

#initialization
import matplotlib.pyplot as plt
import numpy as np

# importing Qiskit
from qiskit import IBMQ, Aer, assemble, transpile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit.providers.ibmq import least_busy

# import basic plot tools
from qiskit.visualization import plot_histogram

def initialize_s(qc, qubits):
    """Apply a H-gate to 'qubits' in qc"""
    for q in qubits:
        qc.h(q)
    return qc

n = 2
grover_circuit = QuantumCircuit(n)
```

```
grover_circuit = initialize_s(grover_circuit, [0,1])
grover_circuit.draw()

grover_circuit.cz(0,1) # Oracle
grover_circuit.draw()
```



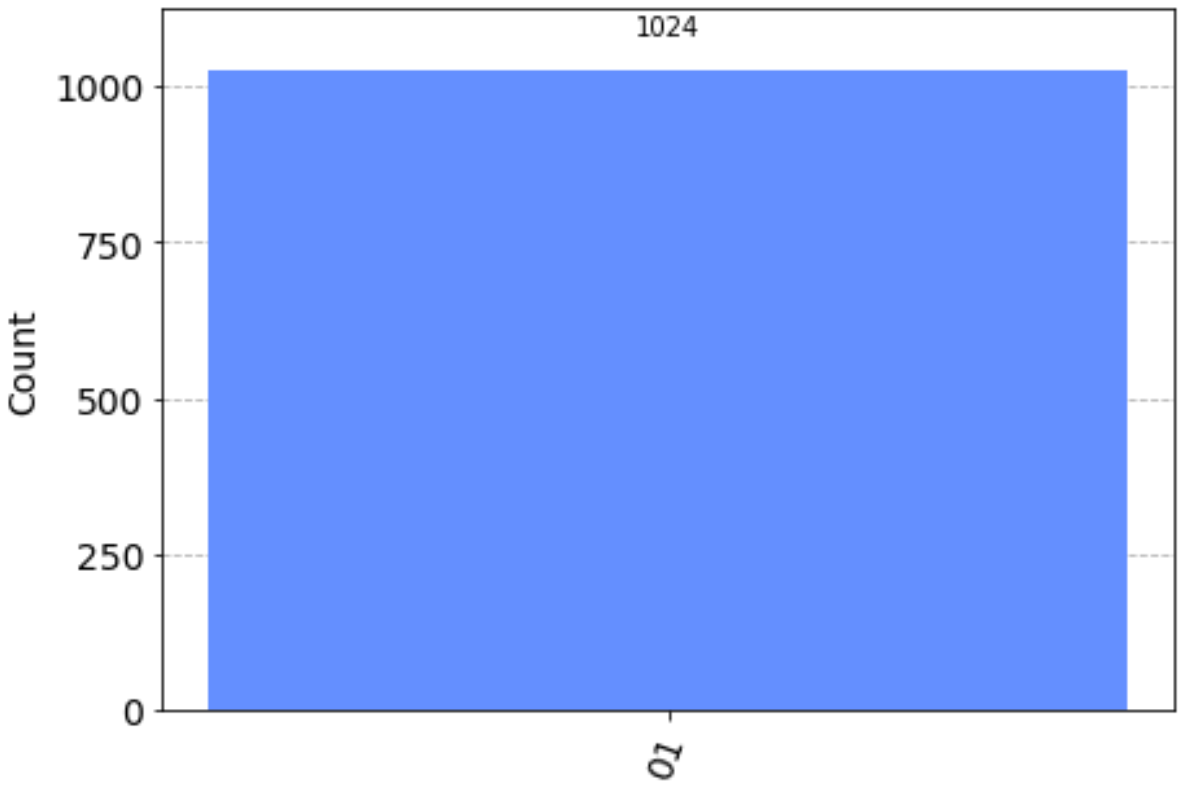

```
# Diffusion operator (U_s)
grover_circuit.h([0,1])
grover_circuit.x([1])
grover_circuit.cz(1,0)
grover_circuit.h([0,1])
grover_circuit.draw()
```




```
sim = Aer.get_backend('aer_simulator')
# we need to make a copy of the circuit with the 'save_statevector'
# instruction to run on the Aer simulator
grover_circuit_sim = grover_circuit.copy()
grover_circuit_sim.save_statevector()
qobj = assemble(grover_circuit_sim)
result = sim.run(qobj).result()
statevec = result.get_statevector()
from qiskit_textbook.tools import vector2latex
vector2latex(statevec, pretext="|\\psi\\rangle =")
```

$$|\psi\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

```
grover_circuit.measure_all()

aer_sim = Aer.get_backend('aer_simulator')
qobj = assemble(grover_circuit)
result = aer_sim.run(qobj).result()
counts = result.get_counts()
plot_histogram(counts)
```




```
# Load IBM Q account and get the least busy backend device
provider = IBMQ.load_account()
provider = IBMQ.get_provider("ibm-q")
device = least_busy(provider.backends(filters=lambda x:
int(x.configuration().n_qubits) >= 3 and
```

```
                                    not x.configuration().simulator and
x.status().operational==True))
print("Running on current least busy device: ", device)

# Run our circuit on the least busy backend. Monitor the execution of the
job in the queue
from qiskit.tools.monitor import job_monitor
transpiled_grover_circuit = transpile(grover_circuit, device,
optimization_level=3)
job = device.run(transpiled_grover_circuit)
job_monitor(job, interval=2)

# Get the results from the computation
results = job.result()
answer = results.get_counts(grover_circuit)
plot_histogram(answer)
```