Grover's Algorithm

Grover's algorithm demonstrates the capability of speed searching from a dataset over its classical counterparts. This algorithm can speed up an unstructured search problem quadratically to the order of $O(\sqrt{N})$. It uses a trick or subroutine for identifying the target number by building an oracle. Whereas, the subroutine to obtain the quadratic runtime improvement is known as amplitude amplification.

As discussed in the lectures, searching one number from a dataset involves the construction of an oracle that identifies the target state from the superposition of qubits. This target state is either marked using an auxiliary qubit with the initial state of $|-\rangle$ in the Boolean/Marked Oracle, or using a phase change in the Phase Oracle. Following this, the amplitude amplification subroutine is used by building a diffuser that increases (amplifies) the amplitude of the target state marked by the oracle constructed in the previous stage.

In this assignment, our task is to find 2 target numbers from a given dataset of N numbers that can be represented using $\log_2(N)$ qubits. Finding two numbers from a similar dataset is similar to the above strategy. However, instead of one oracle, we construct two oracles that identify the two target states. This, in the case of Boolean Oracle, marks two target states on the auxiliary qubit, and in the case of a Phase Oracle adds phase to both the target states in the dataset. The amplitude amplification routine remains as it is and the number of iterations is proportional to $\frac{\pi}{4}\sqrt{\frac{N}{M}}$, where N is the total number of items in the dataset and M is the number of target states.

Here, our task is to find $M = 2$ target states - $|010\rangle$ and $|101\rangle$ - from a dataset of 8 numbers (from 0 to 7). These $n = 8$ numbers can be represented by a total of $\log_2(8) = 3$ qubits on a quantum computer. The oracle used in this implementation is the Boolean/Marked Oracle that uses an additional qubit called an auxiliary qubit for the purpose of marking the target states.

Learning Outcomes of the assignment:

1. Building oracles for a specialized function. The procedure here can be used in various other algorithms like Deutsch-Jozsa, or while building a QRAM (Quantum RAM)[1]

1. Using the technique of amplitude amplification to manipulate the amplitudes and change the success probability of the desired state. One another algorithm based on this technique is Quantum Counting[2].

   Can you answer how many target states M at max can be found from a given dataset of N numbers using Grover's algorithm? Hint: What happens when $N = 2M$? Refer this discussion on Stack Exchange for more hints!

1. Using techniques here in the development of Amplitude Estimation based quantum

algorithms.

References:

1. https://uwaterloo.ca/institute-for-quantum-computing/news/new-questions-about-many-queries-quantum-computing
2. https://en.wikipedia.org/wiki/Quantum_algorithm#Algorithms_based_on_amplitude_amplification
3. https://medium.com/swlh/exploring-the-mechanics-of-quantum-computing-algorithms-179209ac8715

In [1]:
```
# Imports
from qiskit import QuantumCircuit, QuantumRegister
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_histogram
from amp_utils import plot_amplitude
from qiskit.circuit.library.standard_gates import C3XGate
from qiskit.circuit.library.standard_gates import ZGate
```

Initialization of qubits

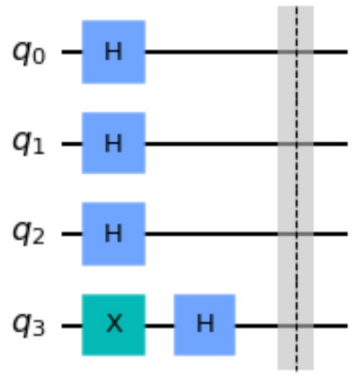
For 3 qubits and 1 anciliary qubit, your initialization should have the given state:

$$|\phi\rangle = H_3 X_3 H_2 H_1 H_0 |0000\rangle$$

NOTE: Index of ancilary qubit is 3

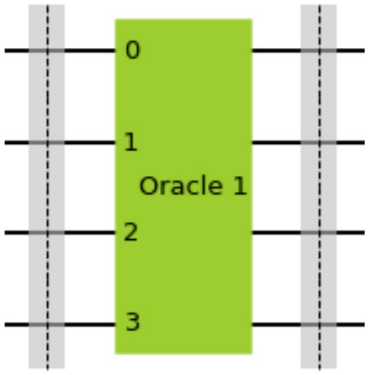
In [2]:
```
# Initialization of qubits

qubits = 3
qc = QuantumCircuit(qubits + 1)    # One additional qubit for ancilary

# WRITE YOUR CODE HERE
qc.x(3)
qc.h(3)
qc.h(2)
qc.h(1)
qc.h(0)

qc.barrier()
qc.draw('mpl')
# WRITE YOUR CODE HERE
```

Out[2]:



Oracle 1 for finding the FIRST target state $|010\rangle$

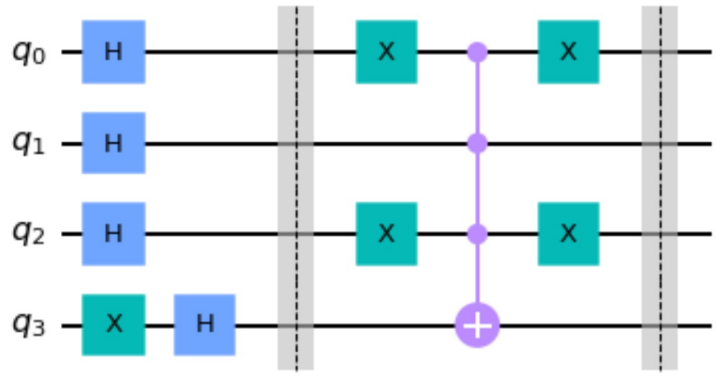
To find the first number, your circuit should look something like this:



In [3]:

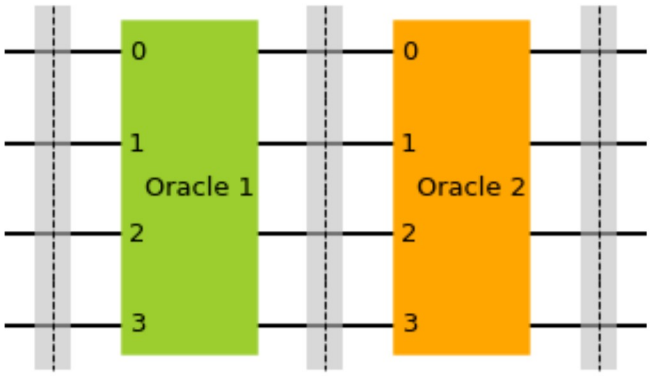
```
# Oracale 1: Make oracle to find only the first number

# WRITE YOUR CODE HERE
qc.x(0)
qc.x(2)
cccxgate = C3XGate()
qc.append(cccxgate,[0,1,2,3])
qc.x(0)
qc.x(2)
qc.barrier()
qc.draw('mpl')
# WRITE YOUR CODE HERE
```

Out[3]:



Oracle 2 for finding the SECOND target state $|101\rangle$

To find the second number, your circuit should look similar to the first oracle:



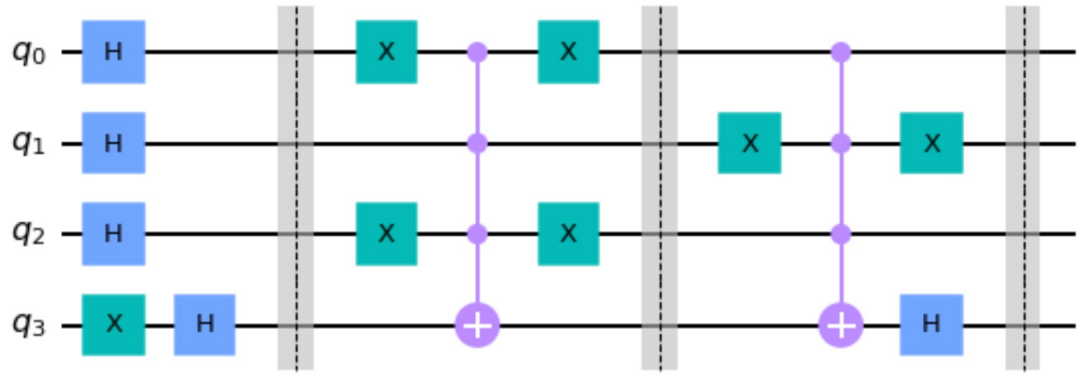
NOTE: We revert the ancilary state back to $|1\rangle$ at the end of the oracle cell to store the negative phase value on the first and second target states.

You DON'T need to add the Hadamard gate. We have done that for you!

In [4]:

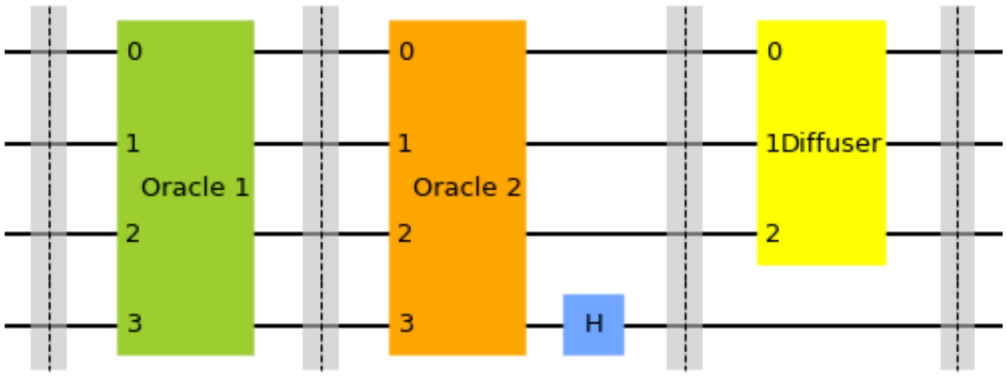
```
# Oracale 2: Make oracle to find only the second number

# WRITE YOUR CODE HERE
qc.x(1)
cccxgate = C3XGate()
qc.append(cccxgate,[0,1,2,3])
qc.x(1)

# WRITE YOUR CODE HERE

qc.h(3)    # Hadamard on ancilary qubit to store the phase value

qc.barrier()
qc.draw('mpl')
```

Out[4]:



Diffuser for Amplitude Amplification

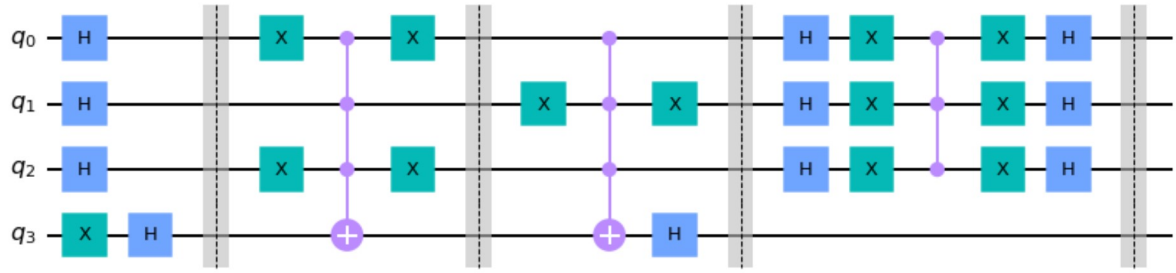
Implement amplitude amplification by constructing a diffuser. Your diffuser should look something like this:



In [5]:

```
# Diffuser for Amplitude Amplification

# WRITE YOUR CODE HERE
qc.h(0)
qc.h(1)
qc.h(2)
qc.x(0)
qc.x(1)
qc.x(2)
cczgate = ZGate().control(num_ctrl_qubits=2, ctrl_state='11')
qc.append(cczgate,[0,1,2])
qc.x(0)
qc.x(1)
qc.x(2)
qc.h(0)
qc.h(1)
qc.h(2)
qc.barrier()
qc.draw('mpl')
# WRITE YOUR CODE HERE
```

Out[5]:

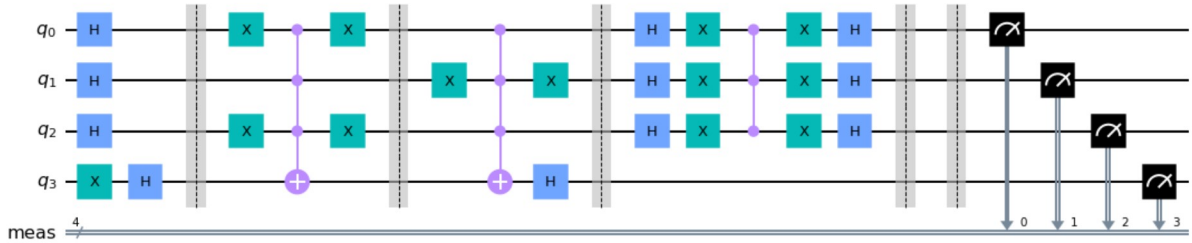


Adding measurements

In [6]:

```
# Measure the qubits

# WRITE YOUR CODE HERE
qc.measure_all()
qc.draw('mpl')

# WRITE YOUR CODE HERE
```

Out[6]:



Running circuit on simulator or quantum backend devices

In [7]:

```
# Imports
from qiskit import Aer, execute, IBMQ
from qiskit.tools import job_monitor
```

In [9]:

```
# Run on a simulator or quantum backend
sim_backend = Aer.get_backend('aer_simulator')

provider = IBMQ.load_account()
qc_backend = provider.get_backend('ibmq_lima')
```

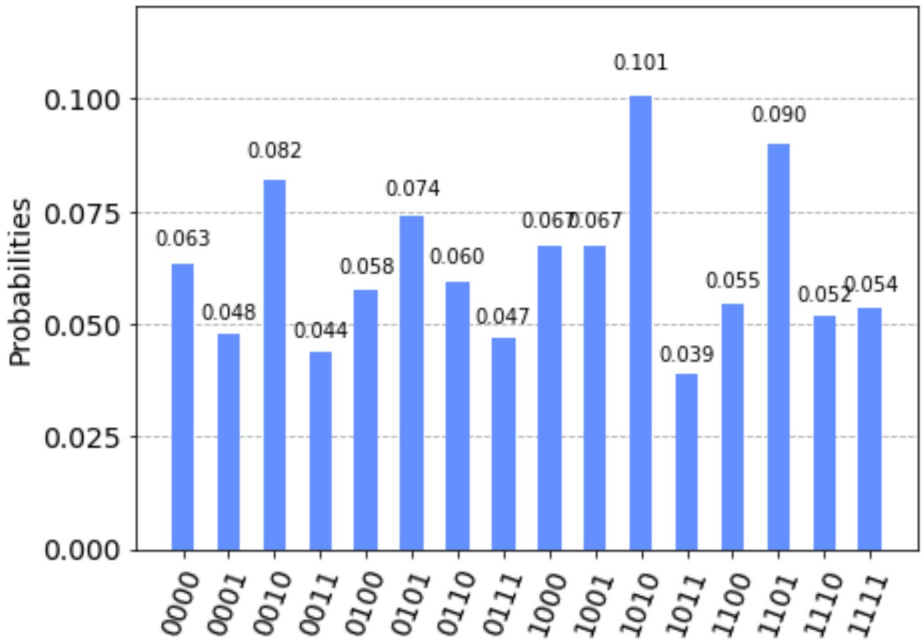
```
ibmqfactory.load_account:WARNING:2022-03-09 23:05:31,409: Credentials are alre
ady in use. The existing account in the session will be replaced.
```

In [10]:
```
# Get counts
job = execute(qc, qc_backend, shots = 1024)
print(job_monitor(job))
result = job.result()
counts = result.get_counts()
print(counts)
```

```
Job Status: job has successfully run
None
{'0000': 65, '0001': 49, '0010': 84, '0011': 45, '0100': 59, '0101': 76, '0110
': 61, '0111': 48, '1000': 69, '1001': 69, '1010': 103, '1011': 40, '1100': 5
6, '1101': 92, '1110': 53, '1111': 55}
```

In [11]:
```
# Visualize the results
plot_histogram(counts)
```

Out[11]:



In [ ]: