Exercises for Introduction to Quantum Computing

Name: Pugazharasu Anancia Devaneyan (s6puanan)
Matriculation number: 3300280

In [ ]:
```python
#Importing the required libraries
import matplotlib.pyplot as plt
import math
import numpy as np
from math import pi
import qiskit as qi
from qiskit import IBMQ, BasicAer, Aer, transpile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, execute
from qiskit.visualization import plot_histogram
from qiskit_ibm_provider import IBMProvider
```

1 Quantum Simulation on Quantum Hardware

a) We know that,

$$e^{-i\omega t A\otimes B} = (\mathbb{I}_n \otimes \mathbb{I}_n)\cdot\cos(\omega t) - i\sin(\omega t)\cdot(A\otimes B)$$

holds true for all

$$A^2 = B^2 = \mathbb{I}_n$$

where $n = dim(A)$. Thus, given the Hamiltonian,

$$H = X_0 \otimes Y_1$$

the unitary representing this time evolution for a small timestep δt is given by,

$$U = e^{-i\delta t H}$$

$$U = e^{-i\delta t(X_0\otimes Y_1)}$$

$$U = (\mathbb{I}_n \otimes \mathbb{I}_n)\cdot\cos(\delta t) - i\sin(\delta t)\cdot(X_0\otimes Y_1)$$

writing this out as matrix elements, we have:

$$U = \cos(\delta t)\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} - i\sin(\delta t)\begin{pmatrix}0&0&0&-i\\0&0&i&0\\0&-i&0&0\\i&0&0&0\end{pmatrix}$$

$$U = \begin{pmatrix}\cos(\delta t)&0&0&-\sin(\delta t)\\0&\cos(\delta t)&\sin(\delta t)&0\\0&-\sin(\delta t)&\cos(\delta t)&0\\\sin(\delta t)&0&0&\cos(\delta t)\end{pmatrix}$$

However, this matrix is not diagonal in the Z basis, thus it is much more convenient to instead write,

$$X_0\otimes Y_1 = H\cdot Z\cdot H \otimes S\cdot H\cdot Z\cdot H\cdot S^\dagger$$

Thus, giving us

$$U = H_0 H_1 S_1 e^{-i\delta t(Z_0\otimes Z_1)} H_0 H_1 S_1^\dagger$$

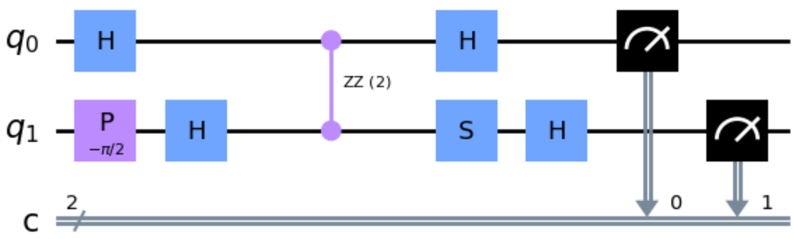
Each of the terms in the above opeartor are diagonal in the computational basis and can be represented as gates. This is our quantum circuit for the given Hamiltonian!

b) We will now implement a quantum circuit on Qiskit for the Hamiltonian from (a)

In [ ]:
```python
def hammy(t):
    t *= 2
    qc = QuantumCircuit(2,2)
    qc.p(-np.pi/2,1)
    qc.h(1)
    qc.h(0)
    qc.rzz(t,0,1)
    qc.s(1)
    qc.h(0)
    qc.h(1)
    qc.measure(0,0)
    qc.measure(1,1)
    return qc
```

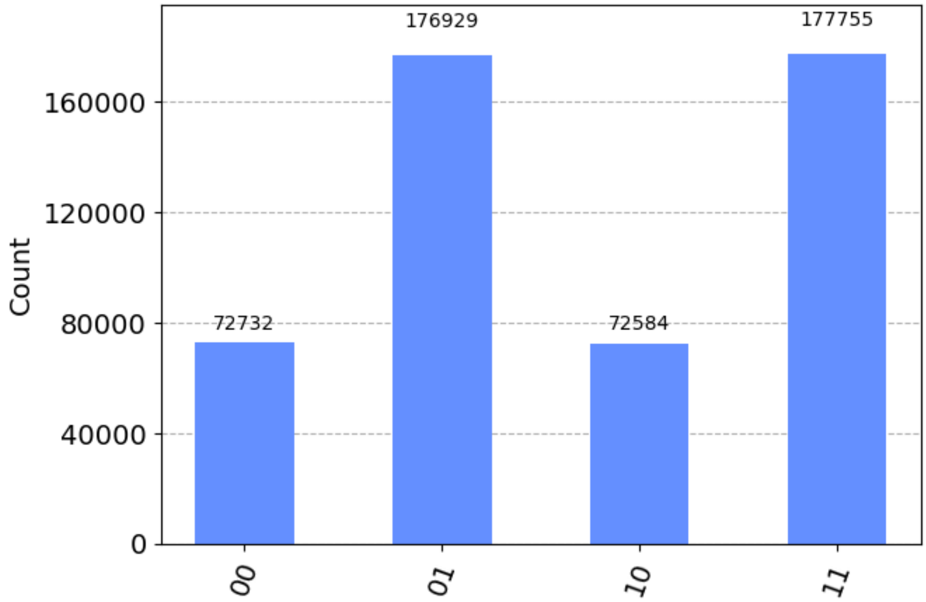
In [ ]:
```python
delta_t = 1
circuit = hammy(delta_t)
circuit.draw('mpl')
```

Out[ ]:



In [ ]:
```python
answer = execute(circuit, backend=BasicAer.get_backend('qasm_simulator'), shots=500000).result().get_counts()
plot_histogram(answer)
```

Out[ ]:



c) We will now attempt to do the same computation in (b) but in a real, noisy quantum computer!

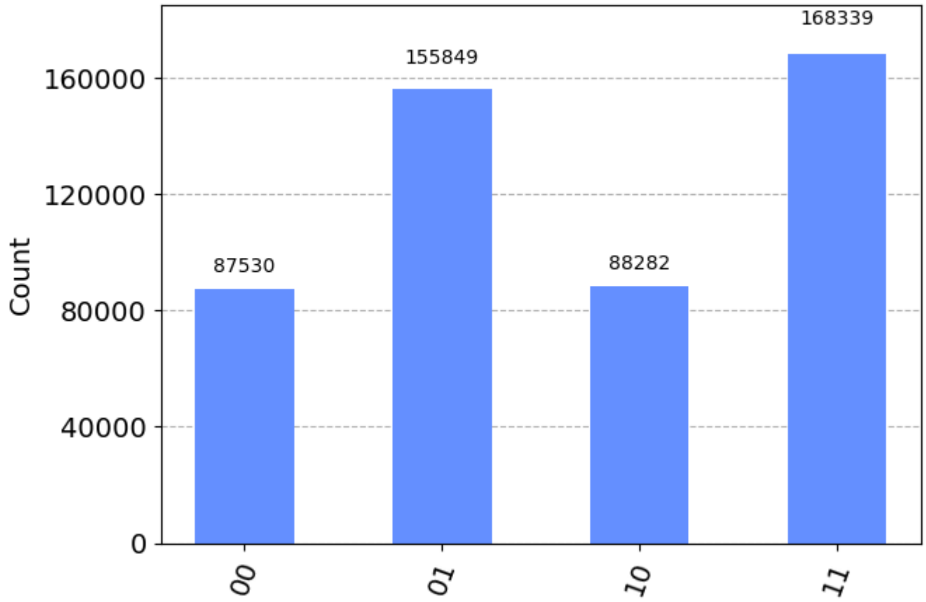
In [ ]:
```python
provider = IBMProvider()
device_list = provider.backends()
for dev in device_list:
    print(dev.name + ':_' + str(dev.configuration().n_qubits) + '_qubits')
```

```
ibmq_lima:_5_qubits
ibmq_belem:_5_qubits
simulator_mps:_100_qubits
simulator_statevector:_32_qubits
simulator_stabilizer:_5000_qubits
ibm_lagos:_7_qubits
ibmq_qasm_simulator:_32_qubits
simulator_extended_stabilizer:_63_qubits
ibmq_manila:_5_qubits
ibm_nairobi:_7_qubits
ibm_perth:_7_qubits
ibmq_jakarta:_7_qubits
ibmq_quito:_5_qubits
```

In [ ]:
```python
num_shots_hardware = 500000
hardware_backend = provider.get_backend('ibmq_belem')
job = execute(circuit,backend=hardware_backend , shots=num_shots_hardware )
```

In [ ]:
```python
counts = job.result().get_counts()
plot_histogram(counts)
```

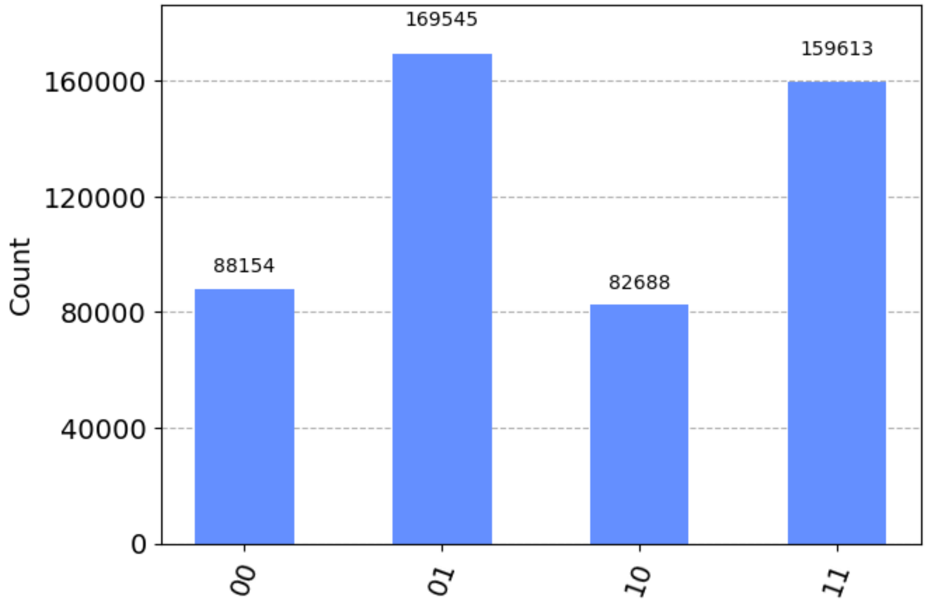
Out[ ]:



We can also use a classical simulator which can simulate the nosie on a quantum device in order to do this computation.

In [ ]:
```python
from qiskit.providers.fake_provider import FakeQuitoV2
num_shots_hardware = 500000
hardware_backend = FakeQuitoV2()
job = execute(circuit, backend=hardware_backend, shots = num_shots_hardware)
counts = job.result().get_counts()
plot_histogram(counts)
```

Out[ ]:



2 Pauli Measurements

a) The unitary transformation such that the final state of the circuit is an eigenstate of the Pauli operator that the unitary is supossed to represent is given by, for the case of the Pauli Z operator,

$$U_1 := Z$$

for the case of the Pauli X operator,

$$U_2 := H$$

for the case of the Paul Y operator,

$$U_3 := H \cdot S^{\dagger}$$

b) Now to generalize from the single-qubit case, we will consider a few cases for two-qubits. The respective unitary transformations U_i for the circuit are given by the table below:

Two-qubit measurements	Unitary
$Z \otimes Z$	$CNOT_{10}$
$\mathbb{I} \otimes Y$	$(H \cdot S^\dagger \otimes \mathbb{I})SWAP$
$X \otimes Z$	$\mathrm{CNOT}_{10}(H \otimes \mathbb{I})$
$X \otimes Y$	$\mathrm{CNOT}_{10}\left(H \otimes HS^\dagger\right)$