Exercício 3 - Física Moderna (ANP)

Aluno: João Vitor Pereira Amorim

Questões:

1. Use estados |+> e medidas repetidas para gerarem números aleatórios. Mostre o código criado por você e gere 6 números de 01 a 60.
2. Como você faria para gerar esses mesmos números com múltiplos qubits?

Soluções:

QUESTÃO 1)

Importação das bibliotecas:

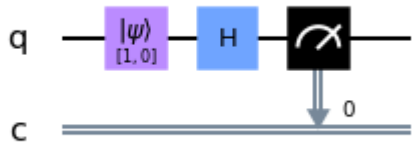
In [59]:
```
%matplotlib inline
import matplotlib
from math import pi
from qiskit import *
from qiskit.extensions import Initialize
qiskit.__qiskit_version__
```

Out[59]:
```
{'qiskit-terra': '0.15.2',
 'qiskit-aer': '0.6.1',
 'qiskit-ignis': '0.4.0',
 'qiskit-ibmq-provider': '0.9.0',
 'qiskit-aqua': '0.7.5',
 'qiskit': '0.21.0'}
```

Inicialização do circuito com **apenas um qubit**, o |+>:

In [60]:
```
circuit = QuantumCircuit(1, 1)
initializer = Initialize([1,0])
circuit.append(initializer, [0])
circuit.h(0)
circuit.measure(0,0)
circuit.draw(output='mpl')
```

Out[60]:



Para realizar a medição do nosso circuito quântico, podemos utilizar a seguinte função:

In [61]:
```
def GetMeasureResult(circuit):
    simulator = Aer.get_backend('qasm_simulator')
    result = execute(circuit, backend = simulator, shots = 1).result()
    return list(result.get_counts().keys())[0]
```

Agora que temos nosso qubit no estado de superposição |+> e a função que fará a medida do qubit, já podemos gerar os 6 números aleatórios entre 01 e 60:

In [65]:
```
numbers = []
quantity_of_numbers = 6
max_number_allowed = 60
quantity_of_bits_needed = max_number_allowed.bit_length()

for i in range(quantity_of_numbers):
    decimal_number = max_number_allowed + 1

    while decimal_number > max_number_allowed:
        binary_number = []

        for i in range(quantity_of_bits_needed):
            result = GetMeasureResult(circuit)
            binary_number.append(result)

        decimal_number = int("".join(x for x in binary_number),2)

    numbers.append(decimal_number)

print(numbers)
```

```
[35, 41, 27, 50, 19, 6]
```

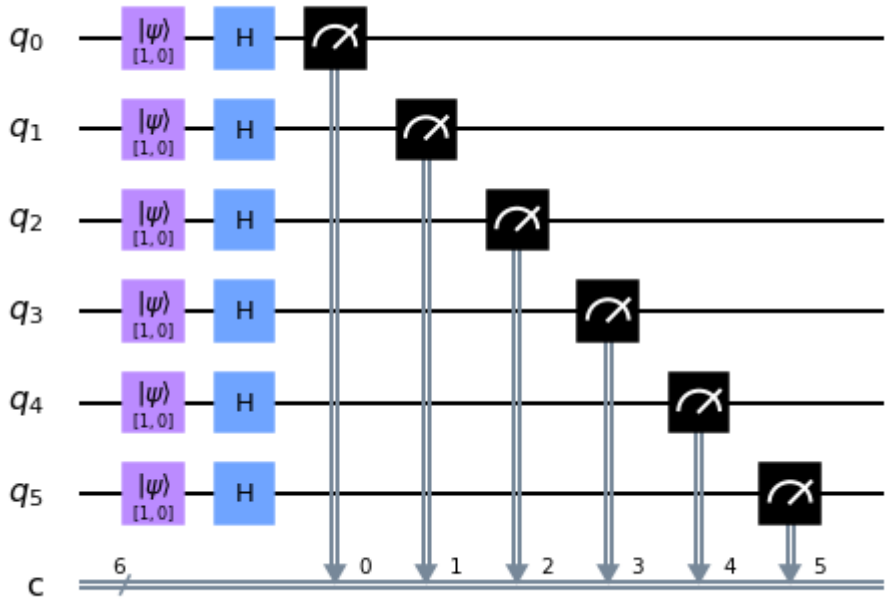
Acima, temos os 6 números aleatórios gerados com **apenas um qubit**.

QUESTÃO 2)

Para gerar os números aleatórios utilizando **múltiplos qubits**, precisamos de um novo circuito quântico com mais qubits no estado |+>:

In [67]:
```
quantity_of_numbers = 6
max_number_allowed = 60
quantity_of_bits_needed = max_number_allowed.bit_length()

circuit = QuantumCircuit(quantity_of_numbers, quantity_of_numbers)

initializer = Initialize([1,0])

for i in range(quantity_of_bits_needed):
    circuit.append(initializer, [i])
    circuit.h(i)
    circuit.measure(i,i)

circuit.draw(output='mpl')
```

Out[67]:



Agora que temos nosso circuito, podemos gerar nossos 6 números:

In [72]:

```
numbers = []

for i in range(quantity_of_numbers):
    decimal_number = max_number_allowed + 1

    while decimal_number > max_number_allowed:
        result = GetMeasureResult(circuit)
        decimal_number = int(result,2)

    numbers.append(decimal_number)

print(numbers)
```

```
[33, 49, 22, 47, 5, 1]
```

Acima, temos os 6 números aleatórios gerados com **múltiplos qubits**.