In [1]:

```
import qiskit                 # Biblioteca utilizada pela IBM para computadores quântico
s
from qiskit import *          # Importamos tudo

qiskit.__version__
```

Out[1]:

```
'0.9.1'
```

Temos inicialmente 2 qubits, onde inicialmente o estado de cada qubit i em um instante n qualquer pode ser dado separadamente por $q_n\left[i\right] = a\left|0\right\rangle + b\left|1\right\rangle$ .Utilizando sua representação vetorial $\begin{bmatrix} a \\ b \end{bmatrix}$, sendo a base do espaço de cada qubit $b_q = \begin{bmatrix} \left|0\right\rangle \\ \left|1\right\rangle \end{bmatrix}$, então o estado inicial de cada qubit é:

- $q_0\left[0\right] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- $q_0\left[1\right] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

In [2]:

```
qc = QuantumCircuit(2, 2) #Criamos um circuito com 2 registradores quânticos e dois reg
istradores clássicos
```

Como a representação matricial da porta de Hadamard é dada por:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Podemos aplicá-la no qubit $q_0\left[0\right]$, e obtermos:

$$q_1\left[0\right] = Hq_0\left[0\right] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

O estado atual de cada qubit agora é dado por:

- $q_1\left[0\right] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $q_1\left[1\right] = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

In [3]:

```
qc.h(0) #Operamos o Gate Hadamard em q[0]
```

Out[3]:

```
<qiskit.circuit.instructionset.InstructionSet at 0x1721e1d9a90>
```

Vamos precisar agora trabalhar com o estado composto por ambos os qubits. Então o estado composto é dado por:

$$E_1 = q_1\,[0] \otimes q_1\,[1]$$

$$E_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\1\\0\end{bmatrix}$$

E a base do estado composto será: $b_c = \begin{bmatrix}|0\rangle\\|1\rangle\end{bmatrix} \otimes \begin{bmatrix}|0\rangle\\|1\rangle\end{bmatrix} = \begin{bmatrix}|0\rangle\,|0\rangle\\|0\rangle\,|1\rangle\\|1\rangle\,|0\rangle\\|1\rangle\,|1\rangle\end{bmatrix}$, que vamos denotar de forma simplificada como $b_c = \begin{bmatrix}|00\rangle\\|01\rangle\\|10\rangle\\|11\rangle\end{bmatrix}$.

A porta CNOT tem sua representação matricial dada por:

$$CNOT = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$$

Então operando no sistema, obtemos o estado final:

$$E_2 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\\frac{1}{\sqrt{2}}\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{bmatrix}$$

In [4]:

```
qc.cx(0, 1)    #Aplicamos o CNOT ao par de qubits
```

Out[4]:

```
<qiskit.circuit.instructionset.InstructionSet at 0x1721e1d9cf8>
```

Logo estado final de nosso sistema é:

$$E_2 = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Ou seka, temos 50% de probabilidade do sistem estar em cada estado ($|00\rangle$ ou $|11\rangle$).

In [5]:

```
# Medimos os estados dos registradores quânticos e guardamos nos clássicos
qc.measure([0,1], [0,1])                            # q[0]->c[0]
simulador = BasicAer.get_backend('qasm_simulator')  # Configuramos nosso simulador
trabalho = execute(qc, simulador)                   # Executamos no simulador
resultado = trabalho.result()                       # Obtemos o resultado
print(resultado.get_counts(qc))                     # Printamos o resultado
```

```
{'00': 520, '11': 504}
```

Rodando na nuvem

In [6]:

```
from qiskit.providers.ibmq import least_busy       # Função para sabermos a máquina mai
s livre
# Nosso Token de acesso
TOKEN = '42d35b8e4d7b201db7d51113852936c2c57fda8ef8dcc96e4057224df14603b00582bc2a165cd8
2e15e9911dc21ac19f3d341d109ca840cfbcbbf773592771c4'
IBMQ.save_account(TOKEN,overwrite=True)             # Vamos salvar nossas credenciais
provedor = IBMQ.load_account()                      # E nos conectamos
```

In [7]:

```
# Buscamos o dispositivo menos ocupado
try:
    livre = least_busy(provedor.backends(simulator=False))
    print("Rodando na máquina: ", livre)
    prosseguir = True
except:
    print("Nenhuma máquina disponível.")
    prosseguir = False

if (prosseguir == True):
    # E vamos rodar e mostrar os resultados efetivamente
    trabalho = execute(qc, livre, shots=1024, max_credits=10)
    resultado = trabalho.result()
    print(resultado.get_counts(qc))
```

```
Rodando na máquina:  ibmq_16_melbourne
{'01': 55, '10': 38, '00': 548, '11': 383}
```

In [ ]:

```

```