In [3]:
# implementation of the DJ algorithm on the simulator and the Melbourne backend

```python
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit, Aer, execute
from qiskit.tools.visualization import plot_histogram
from qiskit.tools.monitor import job_monitor

# if you don't have an account you must create one, get an API token, and enable
# or save the account.
# once the account is saved you just need to use load below.
# see Qiskit terra for instructions.
# IBMQ.save_account('my API token')

IBMQ.load_accounts()

# currently existing backends
# backend = IBMQ.get_backend('ibmq_16_melbourne')
# backend = IBMQ.get_backend('ibmqx4')
# backend = IBMQ.get_backend('ibmqx2')
# backend = IBMQ.get_backend('ibmq_gasm_simulator')

g = QuantumRegister(14)
c = ClassicalRegister(14)
DJ = QuantumCircuit(g, c)

# quantum circuit using qubits 2, 3, 4, 11
# on melbourne for these qubits the coupling map is CX2-3, CX4-3, CX11-3
# we take a balanced function f(x2, x4, x11) = x2 + x4 + x11
# this is a linear function f = <a, x> and the algo can also be viewed as the Be
# rnstein-Vazirani algo to find the vector a

# preparation
DJ.x(g[3])
DJ.barrier(g)
DJ.h(g[2])
DJ.h(g[3])
DJ.h(g[4])
DJ.h(g[11])

DJ.barrier(g)

# function f (oracle)
DJ.cx(g[2], g[3])
DJ.cx(g[4], g[3])
DJ.cx(g[11], g[3])

DJ.barrier(g)

# fourier analysis
DJ.h(g[2])
DJ.h(g[4])
DJ.h(g[11])

# measurement of bits 2, 4
DJ.barrier(g)

DJ.measure(g[2], c[2])
DJ.measure(g[4], c[4])
DJ.measure(g[11], c[11])
```

CalculQuantique-DJalgo-Melbourne-Copy1 http://localhost:8888/nbconvert/html/Desktop/IBMQexperience/te...

1 of 6

20.03.19, 14:19
Out[3]: <qiskit.circuit.measure.Measure at 0x1e2704a8>
In [4]:  #drawing the circuit for verification

DJ.draw()
Out[4]:

```
q2_0: |0>
q2_1: |0>
q2_2: |0>
q2_3: |0>  X  H  X  X  H
q2_4: |0>
q2_5: |0>
q2_6: |0>
q2_7: |0>
q2_8: |0>
q2_9: |0>
q2_10: |0>
q2_11: |0>  H  H
q2_12: |0>
q2_13: |0>
c2_0: 0
c2_1: 0
c2_2: 0
c2_3: 0
c2_4: 0
c2_5: 0
c2_6: 0
c2_7: 0
c2_8: 0
c2_9: 0
c2_10: 0
c2_11: 0
c2_12: 0
c2_13: 0
```

The diagram shows a quantum circuit with multiple qubits and operations such as Hadamard (H) gates and X gates. The circuit appears to manipulate the state of these qubits, possibly as part of a quantum algorithm.
In[11]:  
#simulation finds vector a

backend = Aer.get_backend('qasm_simulator')
job_sim = execute(DJ, backend)
sim_result = job_sim.result()

print(sim_result.get_counts(DJ))
plot_histogram(sim_result.get_counts(DJ))

{'00001011000000': 1024}

Out[11]:

In[12]:  
#experiment
#you can also check for availability and current parameters of the backend before calling it.
#see instructions in Quiskit terra.

backend = IBMQ.get_backend('ibmq_16_melbourne')  # circuit above respects constraints of melbourne device.
shots = 1024  # Number of shots to run the program (experiment); maximum is 8192 shots.
max_credits = 3  # Maximum number of credits to spend on executions.

job_exp = execute(DJ, backend=backend, shots=shots, max_credits=max_credits)
job_monitor(job_exp)
In [13]: # results of experiment

result_exp = job_exp.result()
counts_exp = result_exp.get_counts(DJ)

print(result_exp.get_counts(DJ))
plot_histogram([counts_exp])

Out[13]:

```
{'00001010000000': 184, '00001011000000': 380, '00001000000000': 54, '00000010000000': 61, '00001001000000': 116, '00000001000000': 52, '00000011000000': 110, '00000000000000': 67}
```