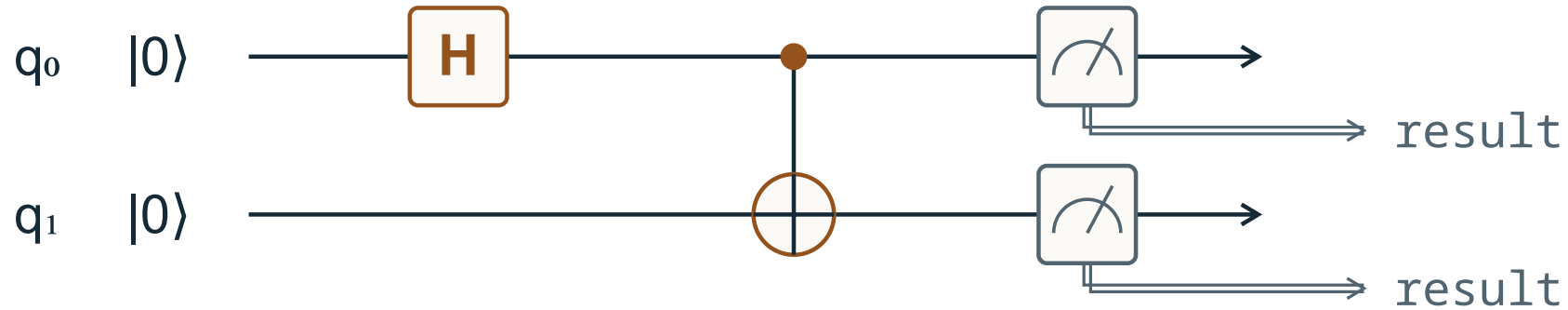


From circuit diagrams to OpenQASM



We need to describe this circuit in a form that quantum software can use.

How do we give the computer these instructions and check what comes back?

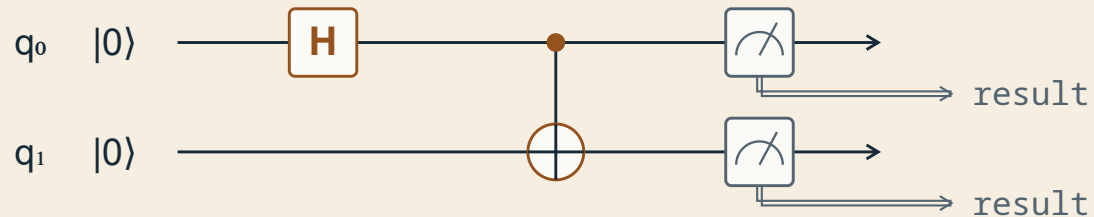
OpenQASM describes the circuit; its measurement records need a statistical interpretation.

- 01 OpenQASM gives quantum software a common description of a circuit.
- 02 Measurement destinations and display order determine what each returned bit means.
- 03 Simulators calculate model predictions that we compare with statistics from repeated runs.

OpenQASM gives different tools a common circuit description.

OpenQASM is an open language for describing and exchanging quantum circuits.

THE CIRCUIT WE WANT TO DESCRIBE



```
OPENQASM 2.0;  
include "qelib1.inc";
```

2.0: the language version used here.
Include: the definitions of gates such as h and cx.

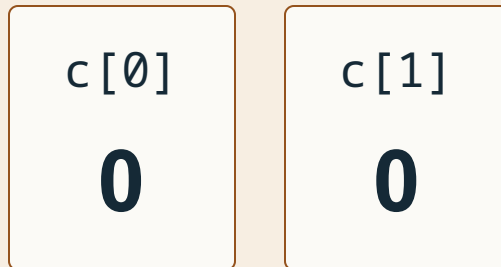
Supporting tools can read the same circuit description and prepare it for execution.

Qubits and classical bits have different jobs.

Q: QUBITS, BEFORE THE FIRST GATE



C: CLASSICAL STORAGE



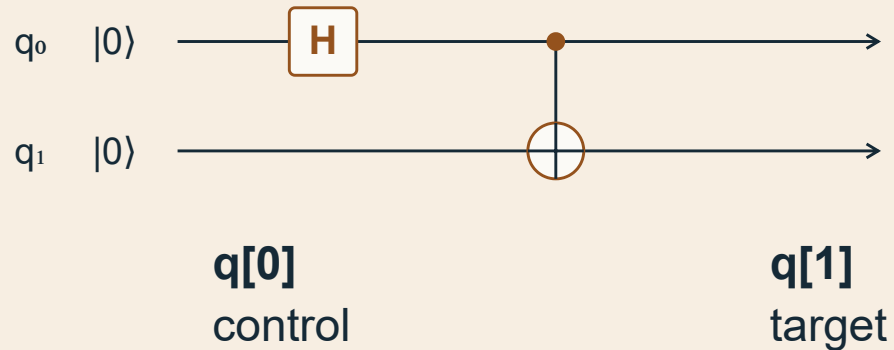
```
OPENQASM 2.0;  
include "qelib1.inc";  
  
qreg q[2];  
creg c[2];
```

Two qubits and two classical bits.
Both registers start at zero in these programs.

Gates act on q. Measurement results are stored in c.

Gate lines specify the qubits and operation order.

OPERATIONS RUN LEFT TO RIGHT



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

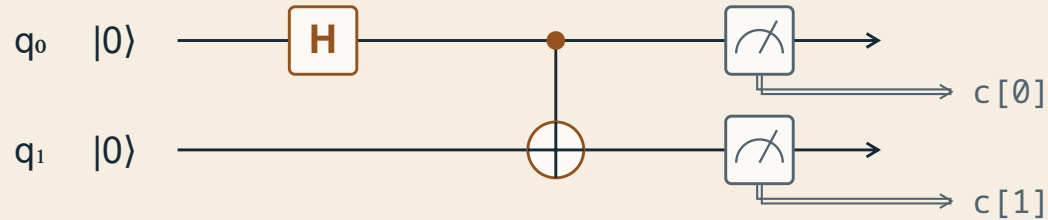
h q[0];
cx q[0], q[1];
```

We read the instructions from top to bottom.
`cx control, target;`

The qubits and the sequence matter as much as the gate names.

A measurement writes an outcome into a classical bit.

MEASUREMENT DESTINATIONS



Each stored outcome is 0 or 1.

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

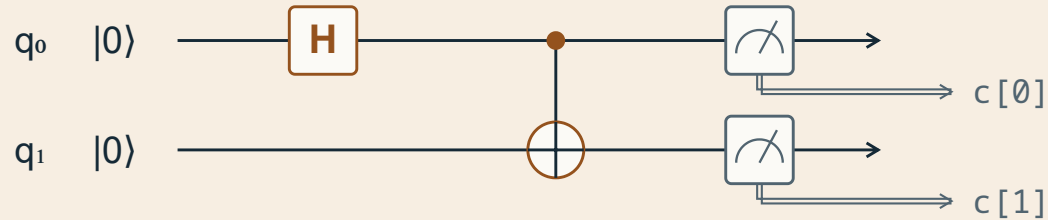
h q[0];
cx q[0], q[1];

measure q[0] -> c[0];
measure q[1] -> c[1];
```

The destination after the arrow tells us where the outcome is stored.

OpenQASM describes the gates and readout.

THE SAME PREPARATION AND READOUT



```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

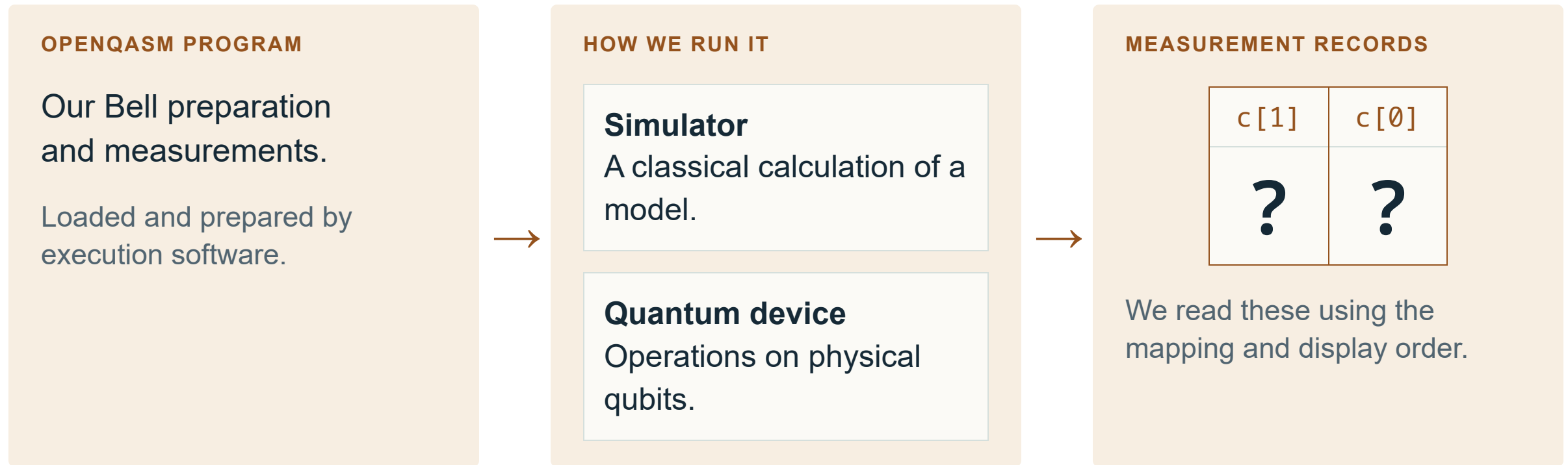
h q[0];
cx q[0], q[1];

measure q -> c;
```

measure q -> c; $\longrightarrow$ measure q[0] -> c[0];
 measure q[1] -> c[1];

For these equal-size registers, the shorthand pairs matching indices.

Execution software runs the circuit and returns measurement records.



Qiskit is a Python toolkit that can load OpenQASM and arrange execution.

The execution setup chooses the simulator or device and the number of runs.

Here, $c[1]$ is printed before $c[0]$.

STORED CLASSICAL BITS



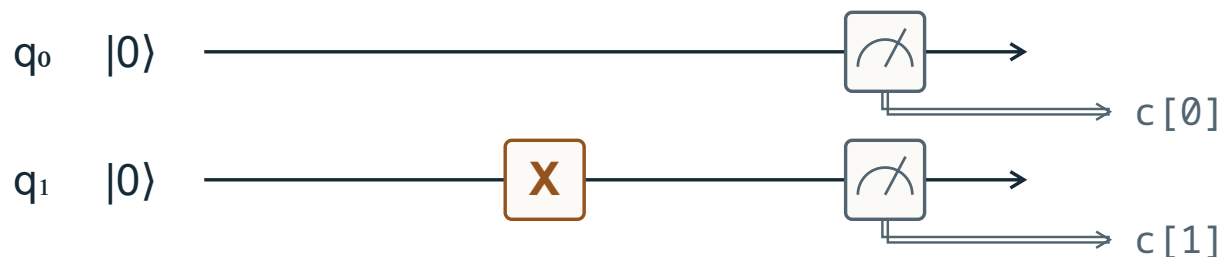
POSITIONS IN THE DISPLAYED STRING

$c[1]$	$c[0]$
?	?

The rightmost digit is $c[0]$. This is a display convention.

Teaching ket: $|q_0q_1\rangle$ · Displayed record: $c[1]c[0]$

We can follow one known result into the display.



QUBIT RESULTS

$$q_0 = 0$$

$$q_1 = 1$$

$$|q_0 q_1\rangle = |01\rangle$$

STORED IN C

$$c[0] = 0$$

$$c[1] = 1$$

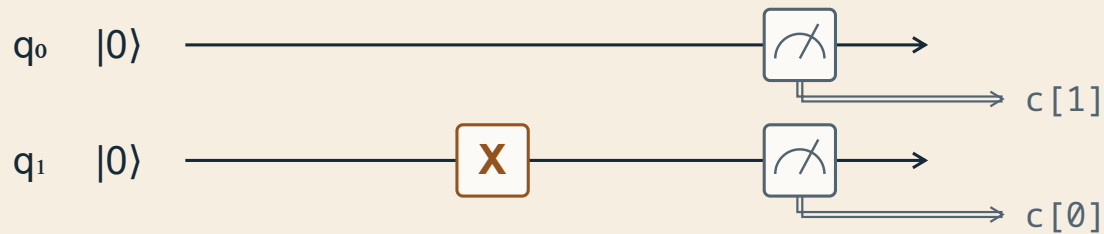
DISPLAYED RECORD

$c[1]$	$c[0]$
1	0

Teaching ket: $|q_0 q_1\rangle$ · Displayed record: $c[1]c[0]$

Which string will we display now?

We keep the same X gate and change only the measurement destinations.



```
measure q[0] -> c[1];
measure q[1] -> c[0];
```

The record is 01.

$q_0 = 0$ is stored in $c[1]$.
 $q_1 = 1$ is stored in $c[0]$.

$c[1]$	$c[0]$
0	1

Teaching ket: $|q_0q_1\rangle$ · Displayed record: $c[1]c[0]$

Unequal test bits make the mapping visible.

Measured results (q_0, q_1)	Matching destinations $q_0 \rightarrow c[0], q_1 \rightarrow c[1]$	Exchanged destinations $q_0 \rightarrow c[1], q_1 \rightarrow c[0]$
(0, 0)	00	00
(1, 1)	11	11
(0, 1)	10	01

The Bell circuit hides this difference. Unequal bits let us check the mapping.

Both output columns use $c[1]c[0]$.

Statevector simulation updates the amplitudes.

A **statevector** is the list of complex amplitudes.

The simulator stores it and applies each gate's linear transformation.

$$|00\rangle \xrightarrow{H \text{ on } q_0} \frac{|00\rangle + |10\rangle}{\sqrt{2}} \xrightarrow{\text{CNOT}} \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

Before measurement	$ 00\rangle$	$ 01\rangle$	$ 10\rangle$	$ 11\rangle$
Calculated amplitude	$1/\sqrt{2}$	0	0	$1/\sqrt{2}$
Predicted probability	$1/2$	0	0	$1/2$

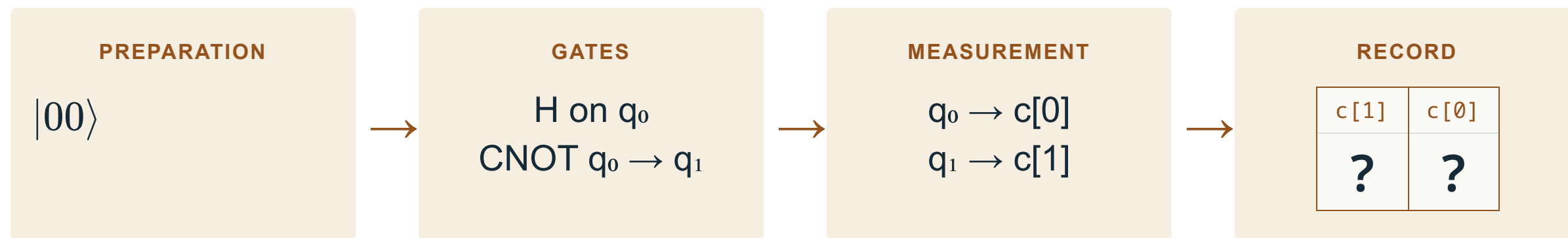
Squared magnitudes give model probabilities.

Sampling them produces simulated measurement records.

For n qubits, this method stores 2^n amplitudes. Basis labels use $|q_0q_1\rangle$.

One shot produces one record.

A **shot** is one complete repetition of this experiment.



Shot 1 **11**

Shot 2 **00**

Shot 3 **11**

Each physical shot starts with a fresh preparation.

Illustrative records from the same ideal Bell circuit.

Relative frequencies estimate measurement probabilities.

100 SHOTS / ILLUSTRATIVE DATA

Record c[1]c[0]	Count
00	46
11	54

$46 + 54 = 100$ records.

OBSERVED FREQUENCY OF 11

$$\frac{54}{100} = 0.54$$

This estimates $P(11)$.

The ideal prediction is 0.5.

A count is a tally. Dividing by the shot total gives a relative frequency.

Does 54 out of 100 mean the circuit is wrong?

IDEAL PREDICTION

$$1/2$$

Probability of record 11.

Expected count in 100 shots: 50.

ILLUSTRATIVE SAMPLE

$$54 / 100$$

Observed frequency: 0.54.

Same circuit and readout.

No. A finite sample can differ from the predicted proportions.

More shots generally improve the estimate. A particular sample need not split exactly in half.

Can sampling alone explain this 01 record?

THE SAME IDEAL BELL CIRCUIT

$$P(00) = P(11) = \frac{1}{2}$$

$$P(01) = P(10) = 0$$

Matching measurement destinations.
Perfect gates and readout.

100 SHOTS / ILLUSTRATIVE REPORT

Record c[1]c[0]	Count
00	46
01	1
11	53

No. Sampling cannot produce an outcome with zero ideal probability.

We need to check the circuit, the recording, the execution model and the reported data.

OpenQASM describes the circuit; its measurement records need a statistical interpretation.

- 01 OpenQASM gives quantum software a common description of a circuit.
- 02 Measurement destinations and display order determine what each returned bit means.
- 03 Simulators calculate model predictions that we compare with statistics from repeated runs.

From the diagram to a program

04.01 · Build the OpenQASM program ↗

01 / REGISTERS

Which qubits and classical bits does the program need?

02 / OPERATIONS

Which gates, operands and order match the diagram?

03 / RECORDING

Where does each measurement result go?

OpenQASM makes the circuit's operations and readout explicit.