Python quantum programming languages

John Scott, Oliver Thomas

Quantum Engineering CDT
University of Bristol

September 19, 2018

Overview

- Python based quantum programming libraries
- We tried to program the common programs (e.g. Grover's algorithm, Shor's algorithm, etc.)
- We tried compiling a simple program for different hardware platforms (i.e. with gate restrictions, etc.)
- We've written a programming guide – we're currently reviewing it

```
# Do quantum stuff
qvm = QVMConnection()
qprog = Program()

# do X on q1, q3, q7
# remember HZH is X
qprog.inst(H(1), Z(1),
 ↪  H(1))
qprog.inst(X(3))
qprog.inst(X(7))
# do measurement over
 ↪  all 8 qubits
for i in range(0, 8):
    qprog.measure(i, i)
```

Short comparison

What is there

- Focussed on quantum circuits
- Apply gates to specific qubits
- Classical control in the same source code
- Python syntax is beginner friendly
- Simulators are available
- Hardware compilers are available

What is lacking

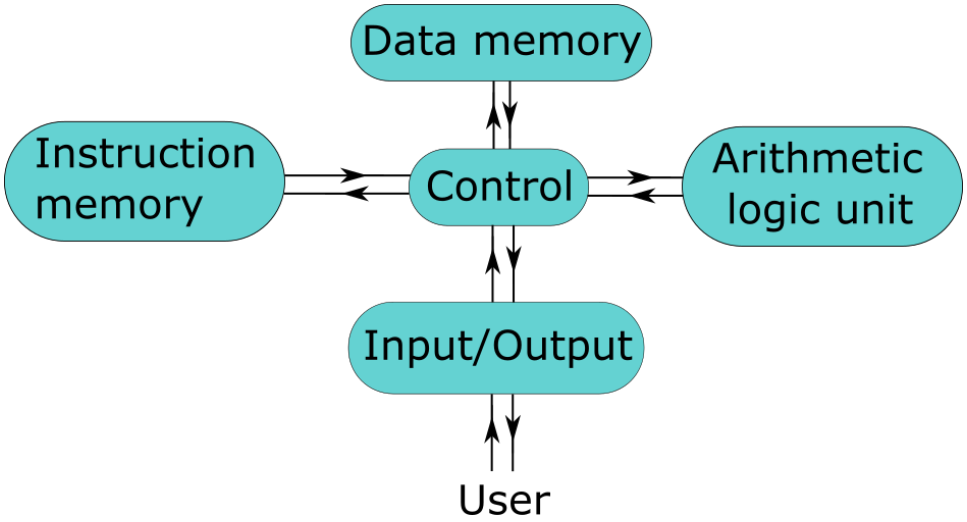
- Lack of support for custom unitaries
- Compilers are not highly developed
- Some languages target specific hardware
- Some simulators are cloud based and require accounts
- No real quantum programming contructs (e.g. quantum if etc.)

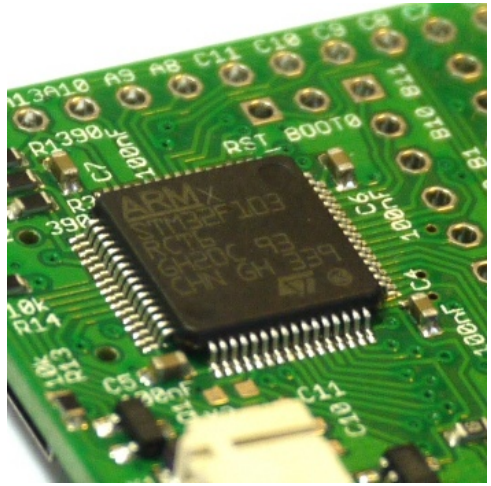
Cloud based quantum computing

- The superconducting qubit developers (IBM, etc.) have made their hardware accessible through the cloud. Part of the purpose of the python programming languages is to connect with this hardware.
- pyQuil has a cloud based simulator, but the connection times out if you try to send large quantum circuits.
- IBM recently introduced their new API [1] which uses JSON files to control runs. They have added pulse shaping. [1]

[1] way too specific and the examples look incredibly confusing

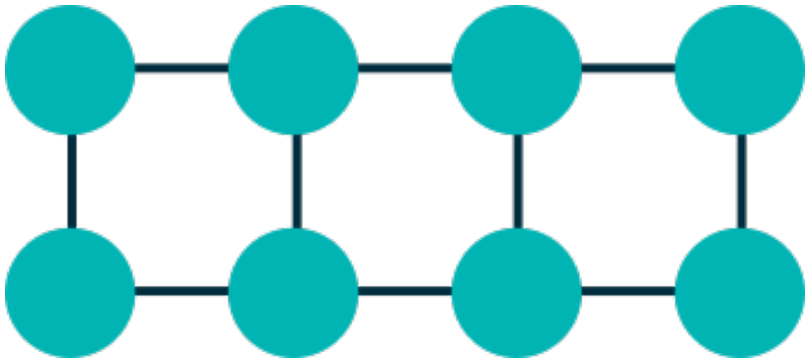
Structure of classical computers

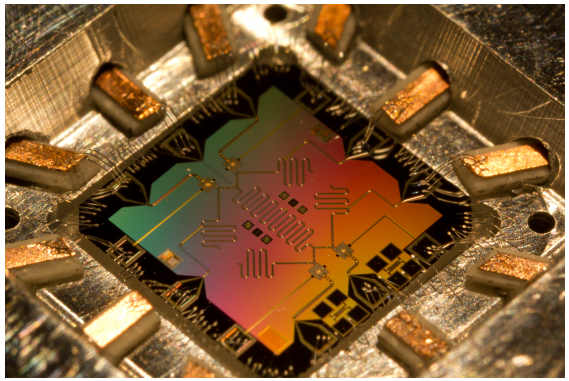




- Classical computers have a lot of internal structure – they are not just a collection of addressable bits.
- Classical computers are controlled using an instruction set, which has elementary operations for arithmetic, control, moving data around, etc.

Structure of quantum computers





- Quantum processing units (QPUs) currently comprise a lattice of qubits which can be manipulated and measured.
- The only ‘instructions’ which can be performed are qubit initialisation, gate operations, and measurement
- Will QPUs eventually involve higher level structures like in the classical case? This will determine the type of languages that will control the devices

Conclusion: long term programming languages

- The structure of long term languages depends on the structure of long term quantum computers
- We need a quantum instruction set that isn't just listing gates. We want to see coherent **addition**, **multiplication**, **exponentiation**, etc. Possible also coherent **branching**?
- Don't see Python being the long term quantum language – there's no need to wrap assembly language in high level syntax.
- Existing Python libraries not built to be scalable languages. Heavy focus on quantum circuits, not useful for new algorithms.

References

[1] David C McKay, Thomas Alexander, Luciano Bello, Michael J Biercuk, Lev Bishop, Jiayin Chen, Jerry M Chow, Antonio D Córcoles, Daniel Egger, Stefan Filipp, et al. Qiskit backend specifications for openqasm and openpulse experiments. *arXiv preprint arXiv:1809.03452*, 2018.