The Future of Common Quantum Assembly Language

Ali Naqvi

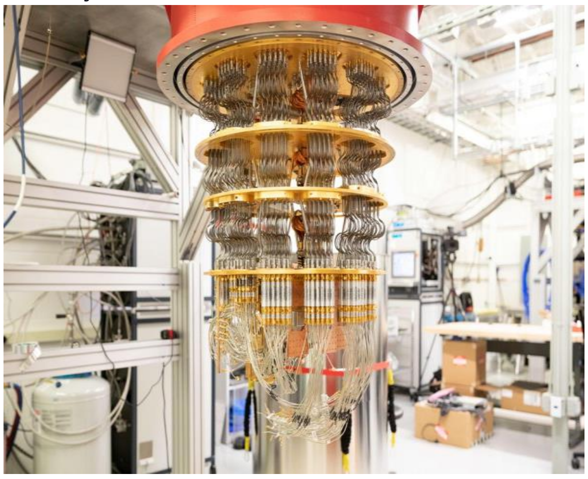
110024562

Computer Architecture II - COMP 2660

2/28/2021

I. Introduction

As human technology progresses, we become more dependent on computers. Because of this, computers are constantly becoming stronger over time. In order to make new discoveries, humans have begun to use quantum computers over conventional computers and supercomputers. Although supercomputers are still dependable, quantum computers have shown more potential. A quantum computer is a device that can perform quantum computation. Its power comes from the ability to manipulate quantum bits also known as qubits. From helping develop novel drugs to creating more dependable batteries for electrical cars, quantum computers are the future. The speed of a quantum computer was also demonstrated by a quantum processor by Google's Sycamore. Sycamore exhibited its power when it finished a task in 200 seconds whereas the world's fastest supercomputer would have approximately taken 10,000 years.



The picture above shows Google's Sycamore which is faster than the world's fastest supercomputer.

It is estimated by the year 2030, quantum computing will be a multi-billion-dollar industry. That said, each individual computer is remarkably expensive. An example can be taken by Sycamore where it sends control signals to the quantum chips using cables and with each of the hundred cables costing $1000 per 2-foot length. Furthermore, to access a quantum computer, the high skillset of quantum programming is also required. The widely accepted program effective in this field is the Common Quantum Assembly Language, also known as cQASM. cQASM first appeared from MIT in 2005 and since then there has been many other quantum languages implemented. From cQASM to Blackbird, most serve the same purpose of inputs or simply just representing quantum circuits. Originally, cQASM was used for rendering images for motives of visualization but has progressed to become a way to specify quantum circuits as an input to a quantum computer. To this day, learning cQASM is a necessity as it is one of the best ways to learn complex problems introduced in quantum computing. Even though quantum languages are vital in the future, a programmer should first learn the basics of a classical computing language such as python or java as well as have a strong background in linear algebra in order to exceed in quantum computing.

II. Importance

With quantum computers on the rise with new research being conducted every day, Common Quantum Assembly Language has become even more vital. Even though we have advanced in the past few years immensely, some problems are still unsolved. Problems like creating an automatic translator or artificial intelligence which is indistinguishable from humans. In 2019, an internet leak to the website of the NASA laboratory created a milestone for quantum computers. It was revealed that Google had used a quantum computer to perform a calculation that even the world's best conventional supercomputer would not be able to achieve. This milestone marked the dawn of quantum computing and established how important CQASM is in the future. Some advantages a quantum computer holds over other types of computers are the ability to solve complicated optimization problems, improving encryption technology by generating random values, creating stronger machine learning systems and simulating real physics of molecular-scale materials. Real life examples of these can be the ability to calculate the shortest time using the least energy on how to deliver packages, distinguishing a fake political video from a real one, and creating more efficient solar panels or electric car batteries. Furthermore, with constant new upgrades to computers, over time we will see higher abstractions of cQASM as well. cQASM describes relatively simple circuits, so in the future years when there will be billions of qubits in quantum computers, this might be an issue. This being said, cQASM is still a good way to learn the fundamentals and will pass down its implementation to some extent to future programs. The way cQASM uses quantum physics to solve problems is a building block for the many other quantum programs to come. With this, we can begin to understand how powerful cQASM can and will be in the future and its importance in research and discoveries.

III. Technological details

In order to fully understand cQASM's syntax, a person would have to fully grasp concepts like what a qubit is and its addressing, measurement registers, quantum gates, and quantum circuits. Qubits, which are exclusive to quantum computers are subatomic particles like electrons or photons that are used similarly to bits on conventional computers. They hold two basis qubit values which are written as |0> and |1>. However, unlike bits, a qubits state can be in a linear combination of both |0> and |1>. This combination is also known as a superposition. Once the number of qubits is defined, we can identify the individuals similar to lists in classical computing languages; "q[i]", where i is {0….N-1] qubits. Even though single qubits are interesting and easy to use, they hold no computational advantage individually. As previously mentioned, we hold single qubits in 2D vectors however for the state of multi qubits, we use 4D vectors. For two qubits, the state must be described with four complex amplitudes. Because of this, we store the amplitudes in 4D vectors. Another phenomenon like superposition in quantum physics is entanglement. Entanglement can be described as a “quantum mechanical phenomenon in which the quantum states of two or more objects have to be described concerning each other, even though the individual objects may be spatially separated”. In terms of qubits, when two qubits exist, the special connection between them is revealed through the results of measurements. The measurement registers which are also known as binary registers, is used for storing the outcome of measurements. For this register, the outcome can be a quantum computation result or be used to apply binary-controlled gates to other qubits. It is also important to note that cQASM is case-sensitive. lower-case and upper-case are both equivalents. Like other programming languages, comments can also be added to make the code more readable. These comments must begin with “#” and can be added as an extension to a code or a new line. Another implementation to note are names. For a readable code, naming qubits and measurement outcomes are very significant. Moreover, this is where the keyword “Map” is used. The keyword “Map” follows two arguments to rename a single qubit. The first being the qubit identifier currently being used, and secondly, the additional name. This can be seen in the example below.

```
version 1.0
# define a quantum register of 3 qubits
qubits 3

# rename qubits
map q[0],data
map q[1],ancilla
map q[2],output

# address qubits via their names
prep_z data
prep_z ancilla
prep_z output
cnot  data,ancilla
cnot  data,extra

# rename classical bit
map b[1],error_syndrome
measure ancilla

#apply binary controlled Pauli-X gate
c-x error_syndrome,q[2]
```

This sample code shows the implementation of single qubits.

In the same way, quantum gates are also a critical topic in cQASM syntax. This syntax supports a universal set of quantum gates that include single, two, and three qubits gates. It provides support the CNOT, Toffoli, and SWAP gate which are commonly adopted controlled gates. With this, the cQASM instructions are followed with space and arguments that are separated with commas. These arguments are the identifiers of the qubits involved in the gate. A list of supported quantum gates that can be extended can be seen underneath.

Quantum Gate	**Description**	**Example**
I	Identity	i q[2]
H	Hadamard	h q[0]
X	Pauli-X	x q[1]
Y	Pauli-Y	y q[3]
Z	Pauli-Z	z q[7]
Rx	Arbitrary x-rotation	rx q[0], 3.14
Ry	Arbitrary y-rotation	ry q[3], 3.14
Rz	Arbitrary z-rotation	rz q[2], 0.71
X90	X90	x90 q[1]
Y90	Y90	y90 q[0]
mX90	mX90	mx90 q[1]
mY90	mY90	my90 q[0]
S	Phase	s q[6]
Sdag	Phase dagger	sdag q[6]
T	T	t q[1]
Tdag	T dagger	tdag q[3]
CNOT	CNOT	cnot q[0],q[1]
Toffoli	Toffoli	toffoli q[3],q[5],q[7]
CZ	CPHASE	cz q[1],q[2]
SWAP	Swap	swap q[0],q[3]
CRK	Controlled Phase Shift ($\pi/2^k$)	crk q[0],q[1],k
CR	Controlled Phase Shift (arbitrary angle)	cr q[0],q[1],angle
c-X	Binary-Controlled X	c-x b[0],q[2]
c-Z	Binary-Controlled Z	c-z b[1],q[2]

This table shows different types of quantum gates, their description, and examples.

Learning these topics extensively will help set the foundation on building quantum algorithms easily. The circuit models for quantum computation are ordered sequences of quantum gates, measurement, and resets. Resets can be performed with a combination of single-qubit gates and simultaneous real-time classical computation. The real-time computations are used for identifying whether a desired state through measurements has been created. We then initialize q0 into a state |ψ⟩ that can then be followed by applying single-qubit gates. Single qubits are known as partial measurements while all qubits are known as register measurements. We can recall that every individual qubit is measured in the z-basis using the keyword “measure” and the targeted qubit. A single qubit measurement with all three basis, is allowed by using the instructions: measure_x, measure_y, and measure_z. On the other hand, the programmer can measure the whole quantum register at once using measure_all. The measurement outcome control binary-controlled gates. With the binary measurement outcome, the programmer can then use it to control a quantum operation. However, it will only be executable if the binary value is 1. We can also use multiple measurement outcomes to control a quantum operation. Finally, when the measurement is performed, the users can see the results. To access this feature, the programmer must implement the command “display [a]” which will return the outcome measurement value of the involved qubit. With cQASM, many algorithms are used which are significantly more effective than its classical algorithm. An example of an algorithm can be the Deutsch-Jozsa Algorithm. In this specific algorithm, we are given a hidden boolean function, that takes the input of a string of bits. It then returns either 0 or 1:

$$f(\{x0,x1,x2,...\})\rightarrow 0 \text{ or } 1$$
$$\text{where } xn \text{ is } 0 \text{ or } 1.$$

This function is guaranteed to be constant or balanced. Balanced functions give 0's for half of the inputs and 1's for the other half. Constant functions however, return all 0's or 1's for any input. Another example of a strong algorithm is Grover's algorithm. This algorithm demonstrates to us how potentially powerful quantum computers can be soon. Grover's algorithm can quadratically search up unstructured search problems. It can also perform the amplitude amplification trick. This trick obtains quadratic run time improvements for many other algorithms. Furthermore, the Bernstein-Vazirani Algorithm is an algorithm that is crucial to know as a quantum computing programmer. It is an extension of the previously mentioned Deutsch-Jozsa algorithm. With this algorithm, we can solve more complex problems that the Deutsch-Jozsa Algorithm was limited to. Furthermore, another algorithm to cover is Simon's algorithm. Simon's algorithm was the first quantum algorithm to show the best classical algorithm against an exponential sped-up algorithm in solving a specific problem. Finally, the most popular quantum algorithm is Shor's algorithm. This algorithm factors integers in polynomial time. In comparison, the best-known classical algorithm is known to solve the factors of two prime numbers in superpolynomial time so using Shor's algorithm to compute the factor of big numbers, saves a lot of memory and takes significantly less time. In addition to these, there are many more quantum algorithms and quantum protocols as well. Algorithms and protocols such as the quantum fourier transform, quantum phase estimation, quantum counting, and the quantum transform. In conclusion, cQASM demands the skill to implement and understand many quantum algorithms and protocols where all require a strong base of math, however knowing all these algorithms and protocols, one can begin programming in CQASM.

IV. DISCUSSION

Currently, there are many different types of dialects being used as quantum computing tools. Since there are different syntaxes, this results in a time-consuming process due to the need for translation. The few quantum programs that are currently active are Blackbird, OpenQASM, Quil, and cQASM. Common Quantum Assembly Language although strong, will not be very dependable in the future. The syntax of cQASM supports single qubits and multi-qubits but when future programs demand billions of qubits, it will be difficult. Vectors will have a large runtime as holding such a large scale of qubits will keep the program from being effective resulting in an incapable performance in large projects. Although cQASM will not be as good as other programs for extremely large projects, the program will not completely die out because of its implementation of strong algorithms. Algorithms such as the Grover's algorithm give quantum computers a large edge over classical computers with their fast-searching databases. With a grasp of the theory of cQASM' syntax, a programmer will find it relatively easy to program in a similar environment. That said, to fully understand the theory, programmers should have a strong background in linear algebra since the program works with matrices and vectors. Furthermore, based on the programming environment such as qiskit which is an open-source SDK, the programmer should also have prior knowledge in another language such as python or java. cQASM was created to bring the quantum computing community together. In the same way, all other quantum languages have the same theory. Most follow laws of gates, measurements, and state preparations. That said, one of the major disadvantages of quantum programs is that they follow the laws of quantum physics' restrictions. A quantum language is known to follow laws of probability instead of certainty which makes it distinguishable from a classical computing language. An example we can use is the GPS. The laws of quantum physics forbid showing all properties of an electron. In this case, it will not be possible for a quantum GPS to show location, speed, and direction all simultaneously. However, as a language of probability, the programmer has access to the likelihood of a qubit as a 1 and 0 at the same time. This creates another gap between quantum and computational computers as it allows quantum computers to communicate and process data at new levels. Furthermore, quantum computers if used by the wrong people, can be a threat to high-security encryption. With most people depending on encryption to protect their secrecy, for example, bank information, hypothetically quantum computers can even breach in banks potentially in the upcoming years when there are more technological advances. The consequences of this data breach can be global and catastrophic. Another disadvantage of a quantum computer is that they require an elusive low temperature to process. For it process, it requires a temperature of negative 460 degrees F which is the lowest temperature in the universe and is very difficult to maintain. A quantum computer is also not available to the public eye. Because of the high range price and the errors that can be caused which can likely destroy the system, the computers are restricted from the public. Researchers are still attempting to increase qubits to 70 without causing errors in accuracy. As discussed earlier, algorithms are essential in learning quantum computing, however, this is also a disadvantage. For every new type of computation, quantum computing requires a new algorithm to perform tasks in their environment. This puts in perspective the many algorithms yet to discover in this field. Although cQASM will be outdated in years to come, it demonstrates how far quantum computing can excel and how we as programmers can solve problems that are considered impossible on classical computers.

V. CONCLUSION

With all things considered, programmers of every kind can agree that quantum computing is not only the future but will change how we approach problems and perceive things. With many different quantum languages and more to come, quantum computing in the following years will change how we as programmers can search a database to improving encryption technology. Furthermore, companies like Google, IBM, and Amazon are all pouring billions of dollars into research in quantum computers. A quantum language like cQASM is a building block for a bigger future and will have its implementation of quantum physics carried on into future languages. For programmers new to this, cQASM would be a great language to start off with as its many resources are online and its topics are easier to learn since they are implemented like python's and assembly language's syntax. In conclusion, quantum computing research is a multi-billion-dollar industry and in years to come, it will excel with cQASM transcending to more updated quantum languages.

References

[1] ChoJul, A. (2020, July 9). *The biggest flipping challenge in quantum computing*.Science Mag. http://www.sciencemag.org/news/2020/07/biggest-flipping-challenge-quantum-computing

[2] Ghose, S. (2020, September 17). *Are You Ready for the Quantum Computing Revolution?* Harvard Business Review. http://hbr.org/2020/09/are-you-ready-for-the-quantum-computing-revolution

[3] Giles, M. (2019, January 29). *Explainer: What is a quantum computer? | MIT Technology Review*. MIT Technology Review; MIT Technology Review. http://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/

[4] N, K. (n.d.). *cQASM v1.0: Towards a Common Quantum Assembly Language*. ArXiv.Org. Retrieved February 28, 2021, from http://arxiv.org/abs/1805.09607

[5] Rehman, J. (2020, October 11). *Advantages and disadvantages of quantum computers - IT Release*. IT Release; https://www.facebook.com/pages/It-Release/352552704770668. http://www.itrelease.com/2020/10/advantages-and-disadvantages-of-quantum-computers/.

[6] The Qiskit Team. (2021, February 23). *Defining Quantum Circuits*. Qiskit; Data 100 at UC Berkeley. http://qiskit.org/textbook/ch-algorithms/defining-quantum-circuits.html

[7] Voorhoede, D. (n.d.). *The basics of Quantum Computing*. Quantum Inspire. Retrieved February 28, 2021, from http://www.quantum-inspire.com/kbase/introduction-to-quantum-computing/