# Introducción al Desarrollo de Software Cuántico

José Manuel García Alonso

UNIVERSIDAD DE EXTREMADURA

jgaralo@unex.es


Software Engineering Group QUERCUS

SPILab

# PROGRAMA DEL CURSO

| CONTENIDO | DÍA |
|---|---|
| Introducción a la Programación Cuántica | Martes |
| Primitivas Cuánticas: Estructura de un programa cuántico | Miércoles |
| Aplicaciones: Algunos Algoritmos | Jueves |
| Servicios Cuánticos | Viernes |

Para la aprobación y/o certificado de asistencia al curso los asistentes deberán concurrir al 80% de las clases dictadas (4 clases).
Para la aprobación del curso se realizará un trabajo final de unas 15 hs. extras en tema a acordar con el docente durante la ELI, a entregar en las 2 semanas posteriores al dictado del curso (propuesta para uniformizar las entregas).
El mecanismo de entrega se informará más adelante.


# Introducción al Desarrollo de Software Cuántico

José Manuel García Alonso

UNIVERSIDAD DE EXTREMADURA

jgaralo@unex.es

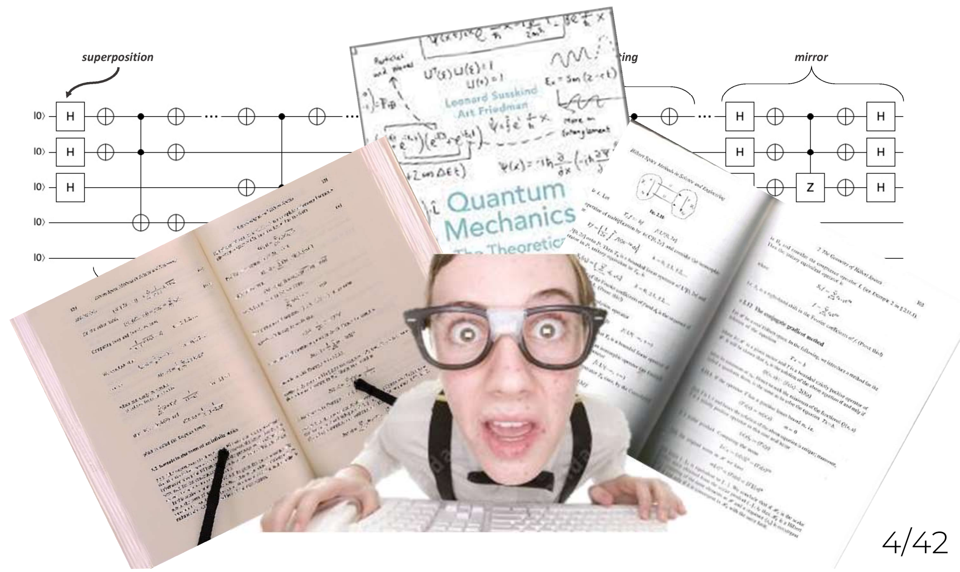
## Introducción a la programación cuántica

Software Engineering Group QUERCUS

SPILab

We are used to making software like driving a Ferrari...

...And this is the next generation Ferrari



# JOURNEY

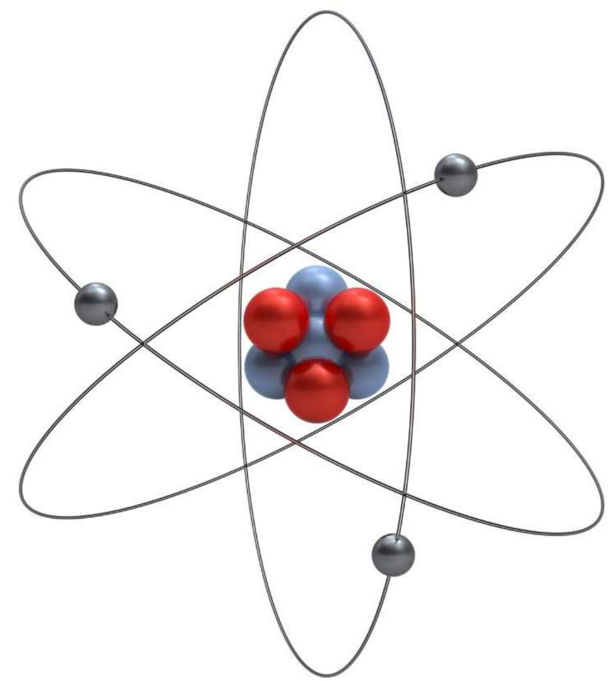
1. Principles of Quantum Computing
2. Quantum vs Classical Programming
3. Quantum Programming

# PRINCIPLES OF QUANTUM COMPUTING

# WHAT

# Prnciples of Quantum Computing. WHAT

Taking advantage of quantum mechanics to do computation, that is, the physical properties of nature at the scale of atoms and subatomic particles



# Principles of Quantum Computing. WHAT

In particular the following principles are used:

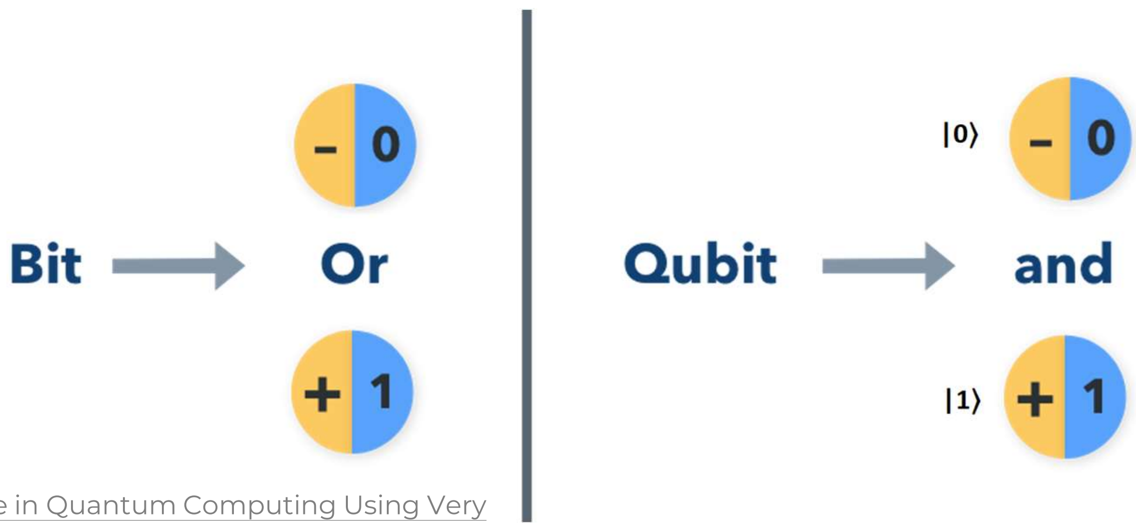
- **Superposition**: being in different states at the same time
- **Entanglement**: State of particles cannot be described independently of the state of the others
- **Collapse**: once observed or measured, the system will always be in one of it possible states

# HOW

# Principles of Quantum Computing. HOW

Using quantum information theory:

Using **Qubit** instead of bit.




$$\alpha \mid 0\rangle + \beta \mid 1\rangle$$

$\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ represent the amplitudes of probability of the qbit

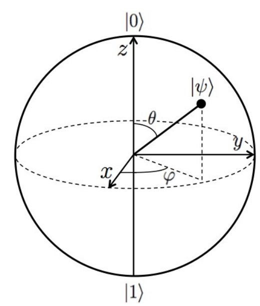
Source: Crash Course in Quantum Computing Using Very Colorful Diagrams

# Principles of Quantum Computing. HOW

## Using Quantum Information Theory:

### QUBIT

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$




$$|\alpha|^2 + |\beta|^2 = 1$$

$$Phase\ 0 \leq \varphi \leq 2\pi$$

$\varphi = \pi$

$|0\rangle$ $\varphi = 0$

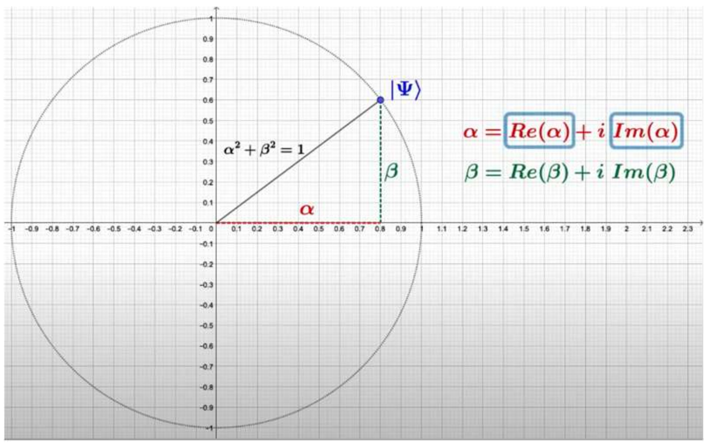
$|1\rangle$ $\varphi = \pi$

**QUBIT amplitude or phase cannot be observed because the system collapse**

### MULTI-QBIT REGISTER

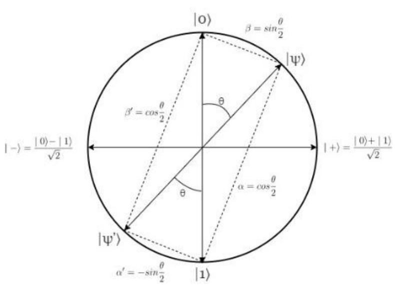
| $\pi$ | $\pi/2$ | $3\pi/2$ | |
|---|---|---|---|
| $\vert 0$ | $0$ | $0\rangle$ | $\varphi = 0$ |
| $\vert 0$ | $0$ | $1\rangle$ | $\varphi = 3\pi/2$ |
| $\vert 0$ | $1$ | $0\rangle$ | $\varphi = \pi/2$ |
| $\vert 0$ | $1$ | $1\rangle$ | $\varphi = 0$ |
| $\vert 1$ | $0$ | $0\rangle$ | $\varphi = \pi$ |
| $\vert 1$ | $0$ | $1\rangle$ | $\varphi = \pi/2$ |
| $\vert 1$ | $1$ | $0\rangle$ | $\varphi = 3\pi/2$ |
| $\vert 1$ | $1$ | $1\rangle$ | $\varphi = \pi$ |

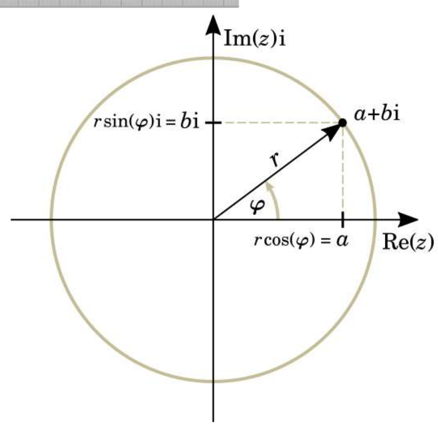
# Introduction to Quantum Computing. HOW




$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

$$|\alpha|^2 + |\beta|^2 = 1$$



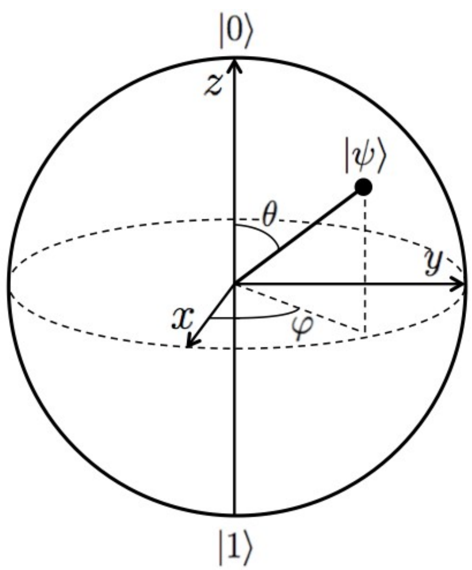

$$|\psi\rangle = (a_\alpha + b_\alpha i)|0\rangle + (a_\beta + b_\beta i)|1\rangle$$

$$|\psi\rangle = r_\alpha e^{i\varphi_\alpha}|0\rangle + r_\beta e^{i\varphi_\beta}|1\rangle$$

$$|r_\alpha|^2 + \left|r_\beta\right|^2 = 1$$

$$|\psi\rangle = e^{i\varphi_\alpha}\left(r_\alpha|0\rangle + r_\beta e^{i(\varphi_\alpha - \varphi_\beta)}|1\rangle\right)$$

# Introduction to Quantum Computing. HOW




This sphere is often called the **Bloch sphere**, and it provides a useful means to visualize the state of a single qubit

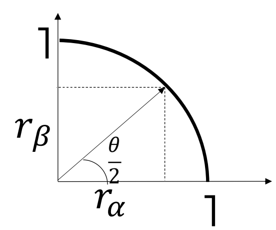
Source: Visualizing Single Qubit Quantum Logic Gates

Irrelevant factor: $e^{i\varphi_\alpha}$

$$|\psi\rangle = e^{i\varphi_\alpha}\left(r_\alpha|0\rangle + r_\beta e^{i(\varphi_\alpha-\varphi_\beta)}|1\rangle\right)$$

$\phi$ = $\varphi_\alpha-\varphi_\beta$

$$|r_\alpha|^2 + \left|r_\beta\right|^2 = 1$$

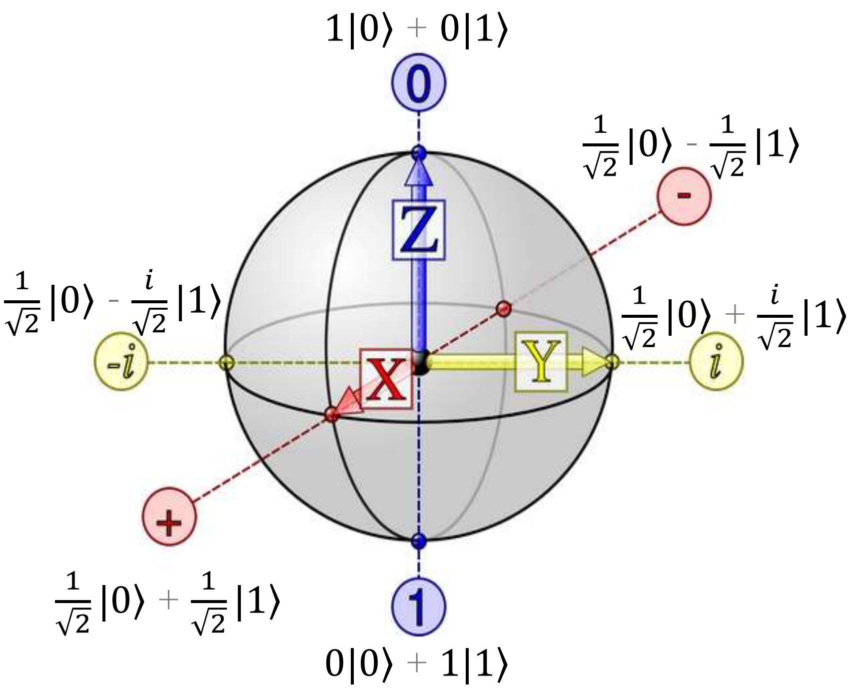



$$r_\alpha = \cos\frac{\theta}{2} \qquad r_\beta = \sin\frac{\theta}{2} \qquad 0 \le \theta \le \pi$$

$$0 \le \varphi \le 2\pi$$

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i(\phi)}\sin\frac{\theta}{2}|1\rangle$$

# Introduction to Quantum Computing. HOW




Irrelevant factor

$$|\psi\rangle = e^{i\varphi_\alpha}\left(r_\alpha|0\rangle + r_\beta e^{i(\varphi_\alpha - \varphi_\beta)}|1\rangle\right)$$

$$|r_\alpha|^2 + |r_\beta|^2 = 1$$

$\phi$

$$r_\alpha = \cos\frac{\theta}{2} \qquad r_\beta = \sin\frac{\theta}{2}$$

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i(\phi)}\sin\frac{\theta}{2}|1\rangle$$

Source; La ciencia de la mula Francis https://francis.naukas.com/2014/07/31/medida-de-la-trayectoria-en-la-esfera-de-bloch-de-un-cubit-superconductor/dibujo20140731-bloch-sphere-qubit-nature-com/

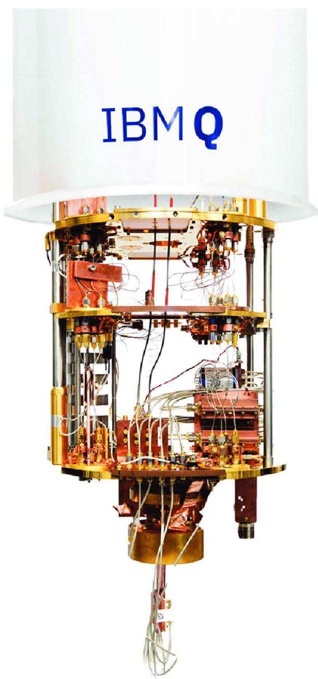
# Introduction to Quantum Computing. HOW

A gentle viasualization of one qbit

https://javafxpert.github.io/grok-bloch/

# Introduction to Quantum Computing. HOW

## To run quantum programs we need:




A device implementing qubits with the ability of initializing them to a known value and manipulating their amplitudes and phases: **Quantum Computer**

Image captured from ResearchGate. https://www.researchgate.net/figure/an-IBM-Quantum-computer_fig1_328723940

A language to communicate with the device and tell it the manipulations that must be performed over the qubits: **Quantum Gates**

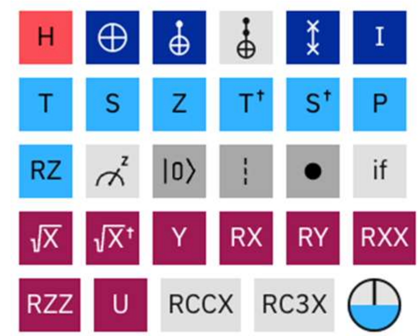
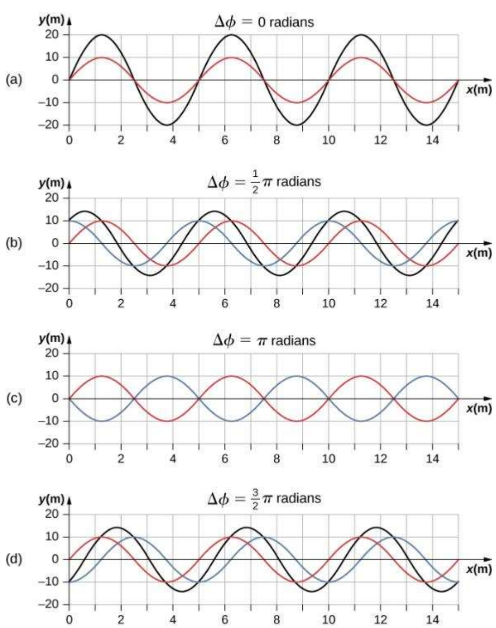



Image captured from IBM Composer. https://quantum-computing.ibm.com/composer/files/new

# Introduction to Quantum Computing. HOW

## Limitations: **QUANTUM DECOHERENCE**





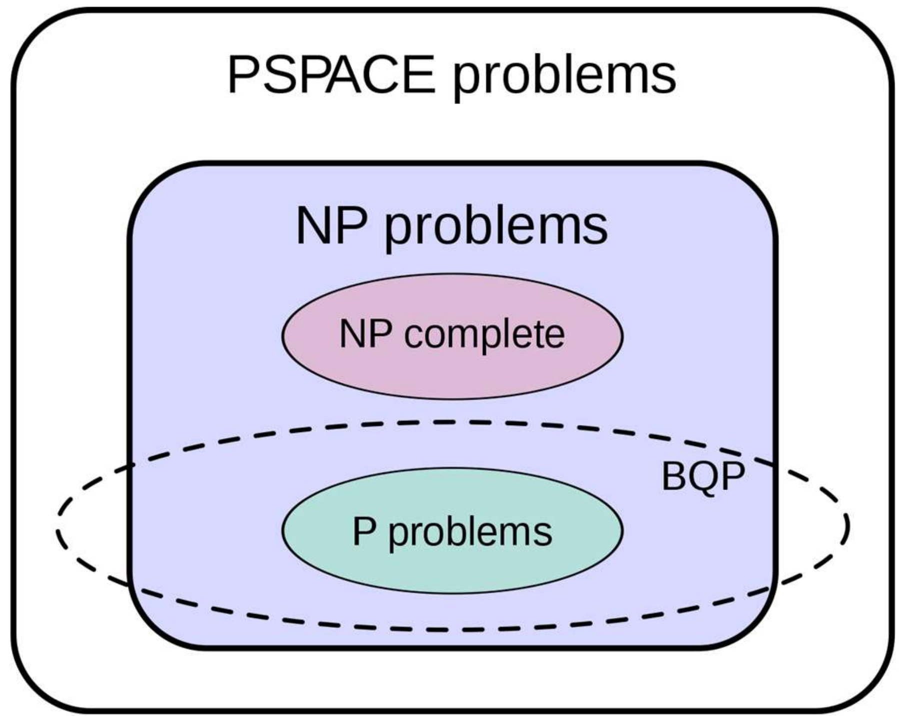

Lost of the phase relation between the states of a quantum computer due to external factors. It introduces errors in the results.

**NISQ Era: Noisy Intermiediate-Scale Quantum**

# WHY

# Introduction to Quantum Computing. WHY

**Bounded-error Quantum Polynomial time** (BQP) is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances




Source: Bounded-error Quantum Polynomial time

# Introduction to Quantum Computing. WHY

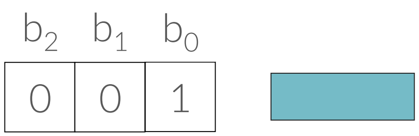
The evidence is that we already know some algorithms:

| Problems | Explication | Quantum Complexity | Classical Complexity |
|---|---|---|---|
| Factorization (Shor) | Decomposition of a number into a product of smaller integers | O(logN) | $\Theta\left(\exp\left(\left(\frac{32}{9}n\right)^{\frac{1}{3}}(\log n)^{\frac{2}{3}}\right)\right)$ |
| Search (Grover) | Search in an unordered sequence of data | O(√n) | O(n) |

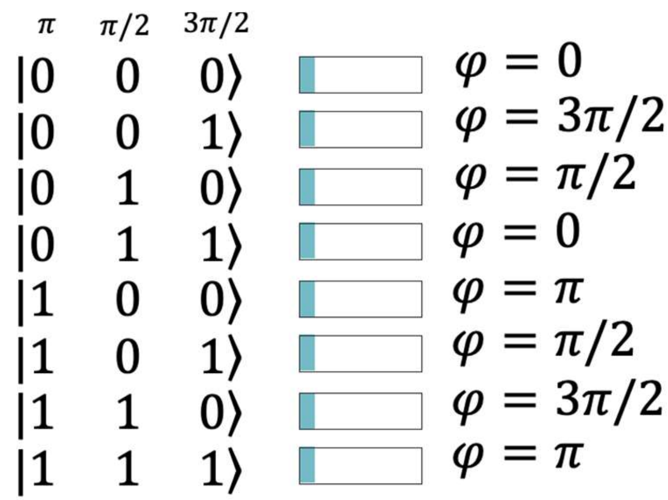
# Introduction to Quantum Computing. WHY

They compute faster because they compute different:

## MULTI-BIT REGISTER




## MULTI-QBIT REGISTER

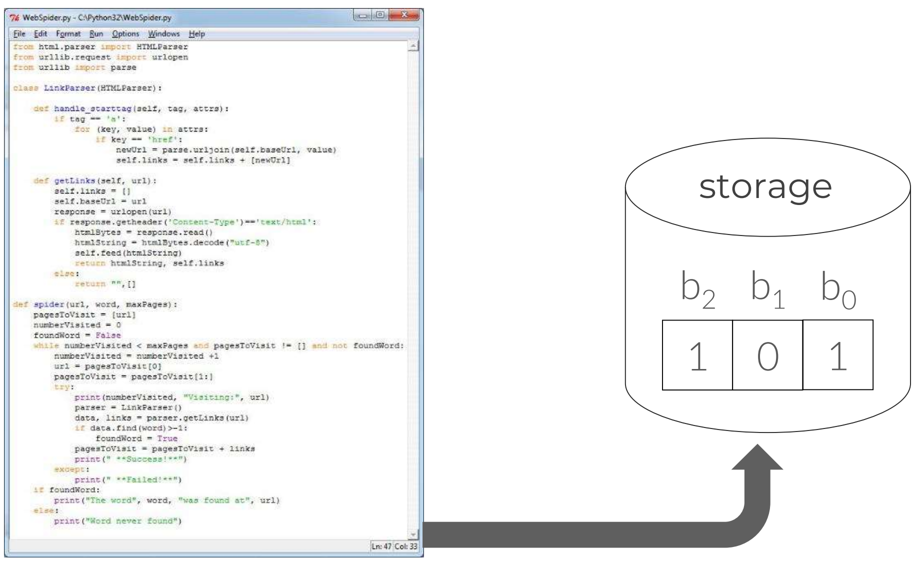


# Quantum vs Classical Programming

# Quantum vs Classical Programming. Difference 1: Strategies

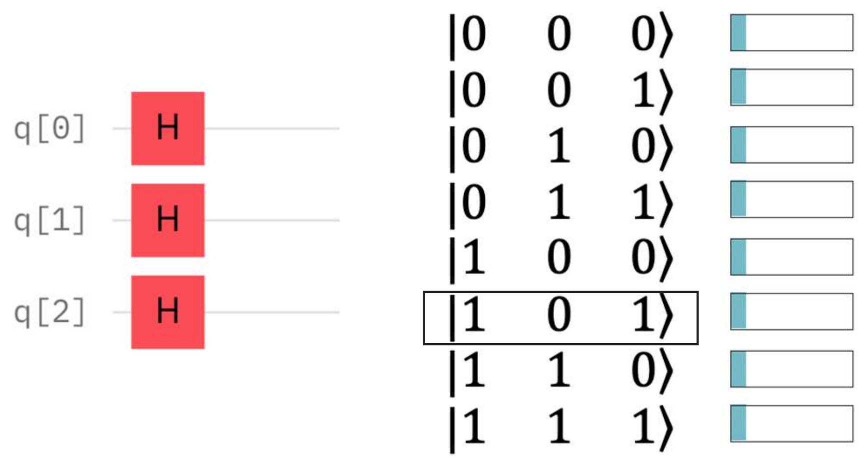
## CLASSICAL

### PRODUCE




## QUANTUM

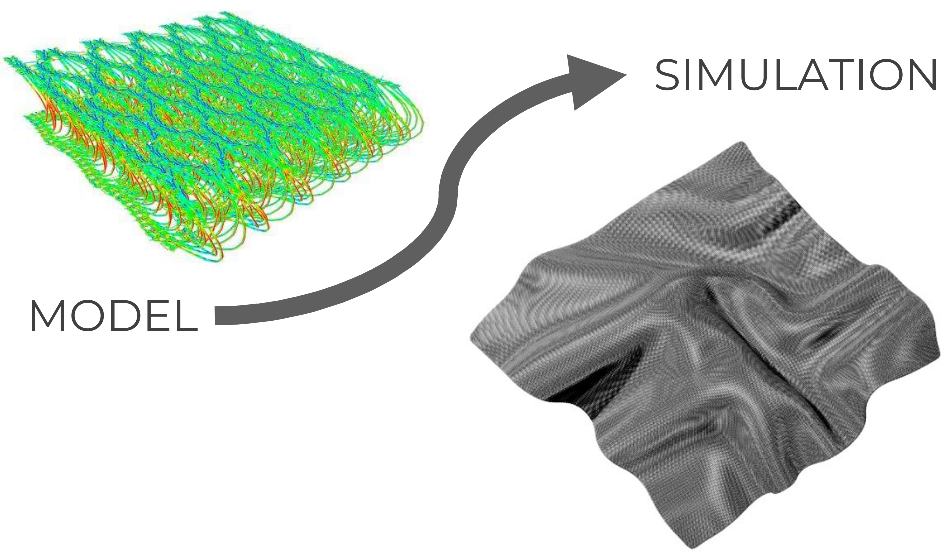
### SEARCH



# Quantum vs Classical Programming. Difference 2: Physical things

## CLASSICAL

### SYNTHETIC




## QUANTUM

### PHYSICAL

# Quantum vs Classical Programming. Structure of a Quantum Program

## A Quantum Program looks like:

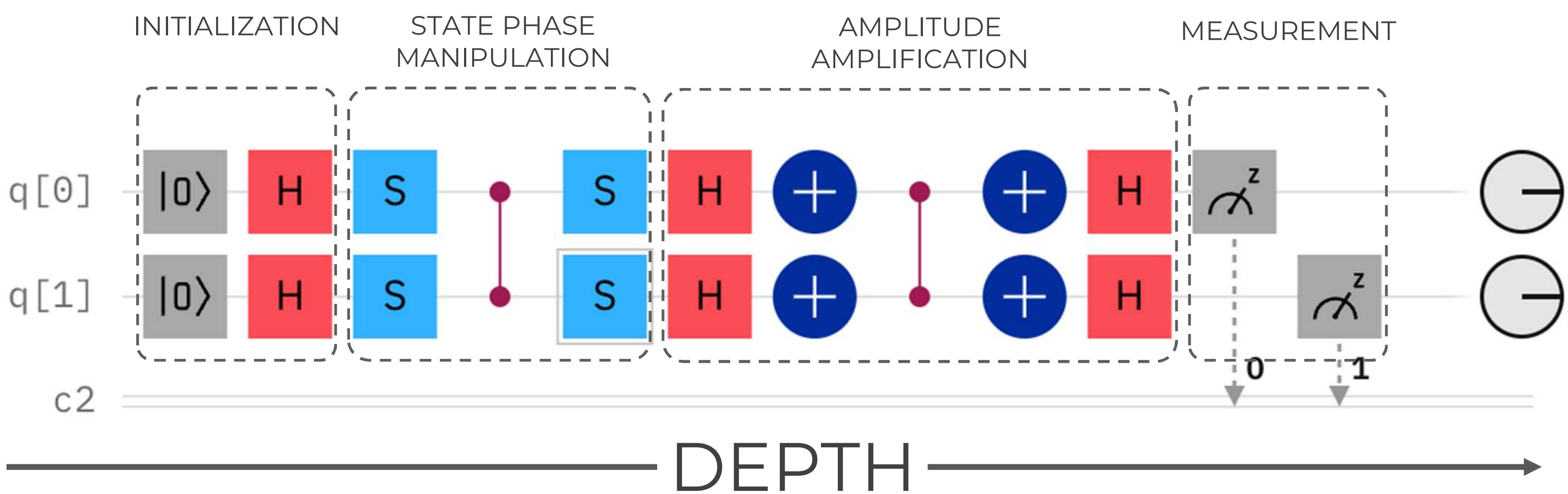


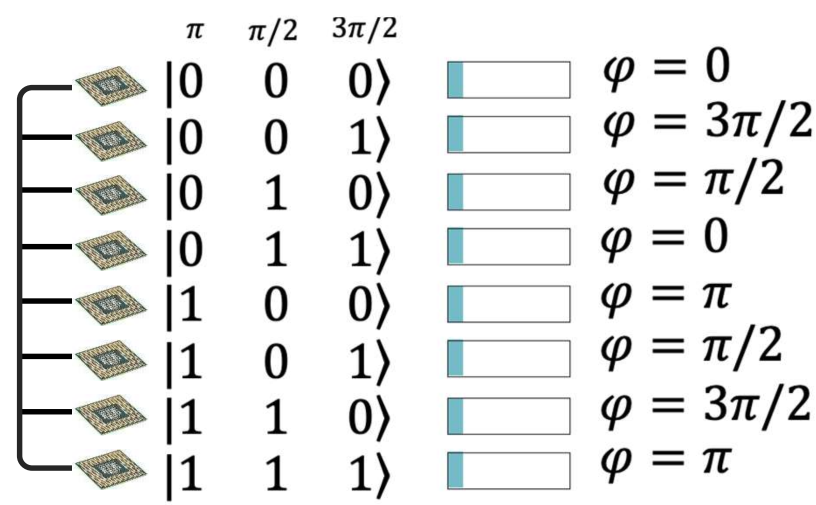

Image from IBM Composer. https://quantum-computing.ibm.com/composer/docs/iqx/example-circuits/grover

# Quantum vs Classical Programming. The real power of Quantum Comp

## They compute faster because they compute different:

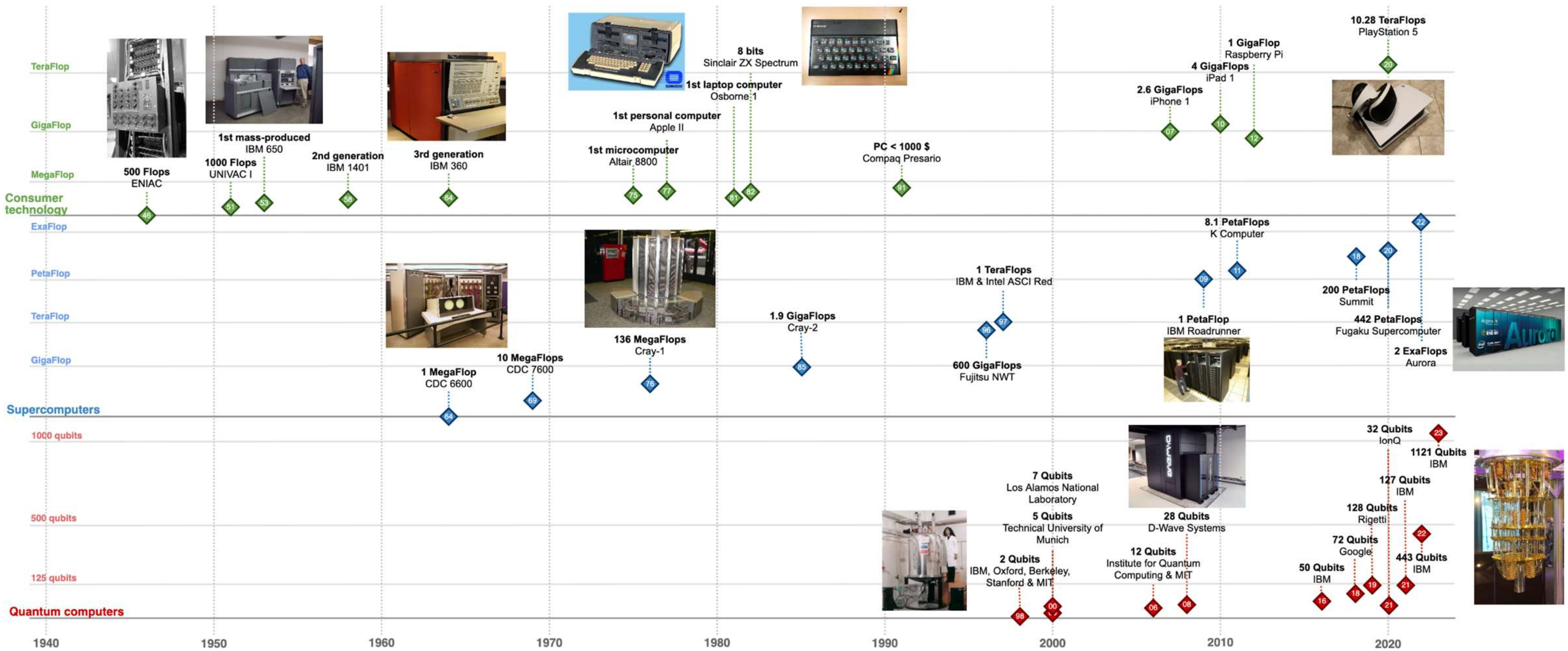
Quantum computing is much more powerful than parallel computing because the superposition states are manipulated as a single one....this "magic" happens when the properties of quantum mechanics are applied to the superposition states

MULTI-QBIT REGISTER

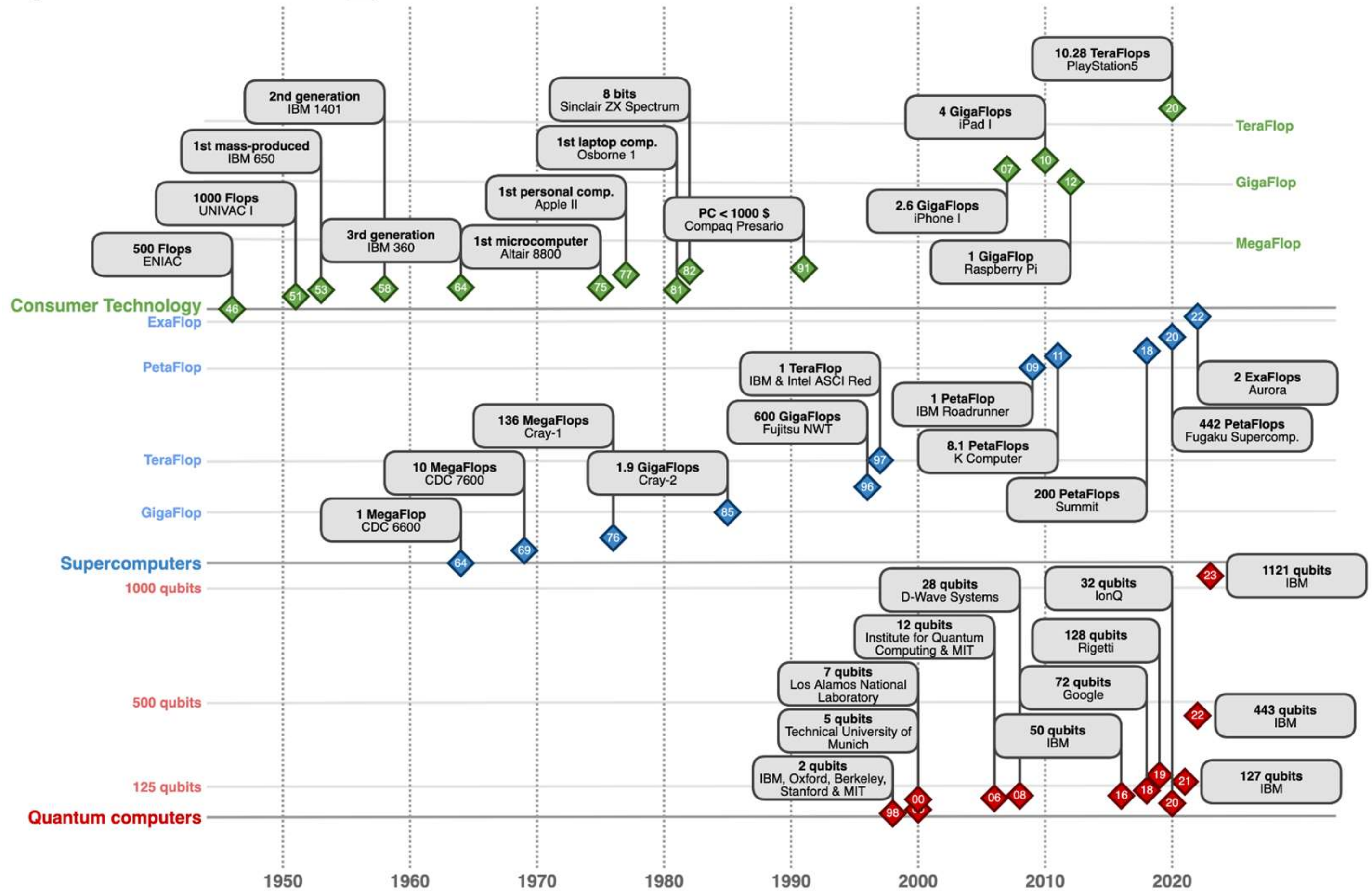


# WHEN

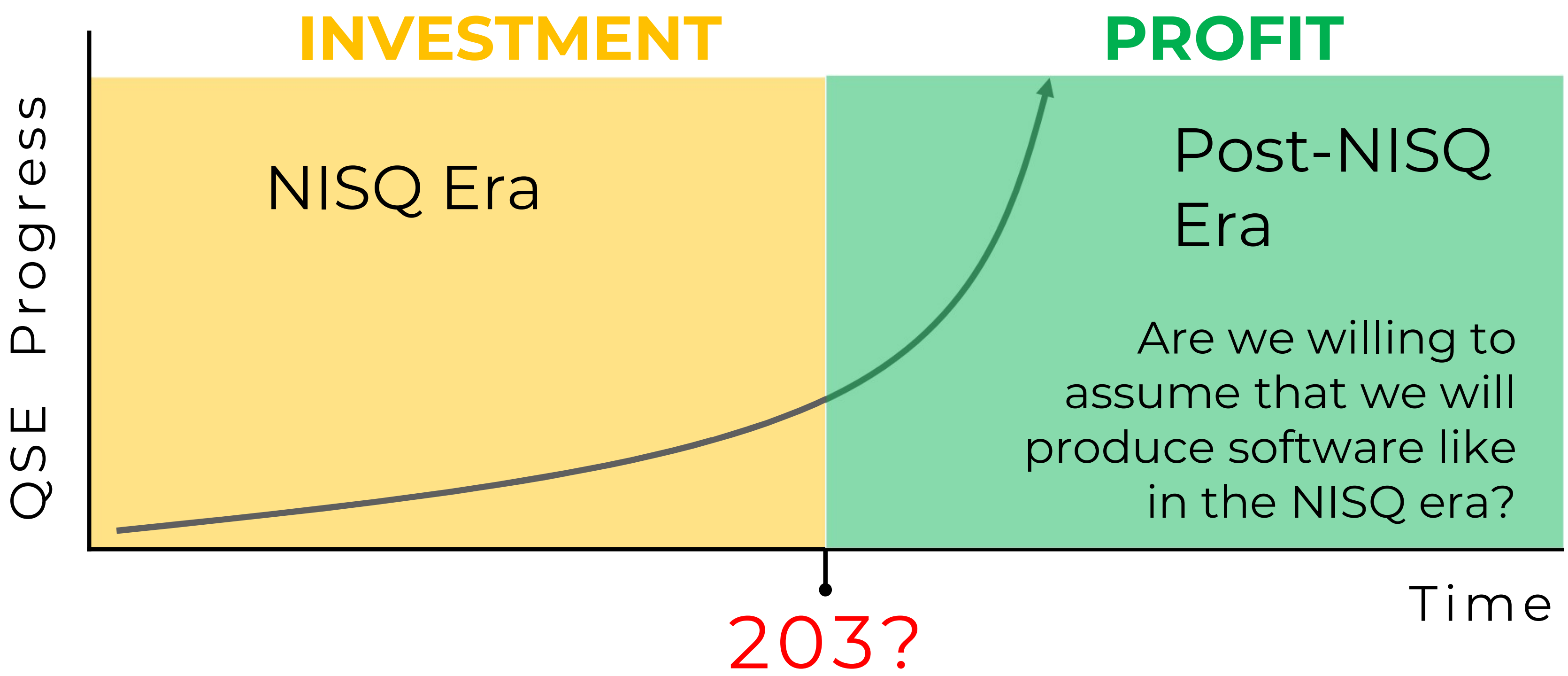
# Introduction to Quantum Computing. WHEN



# Introduction to Quantum Computing. WHEN



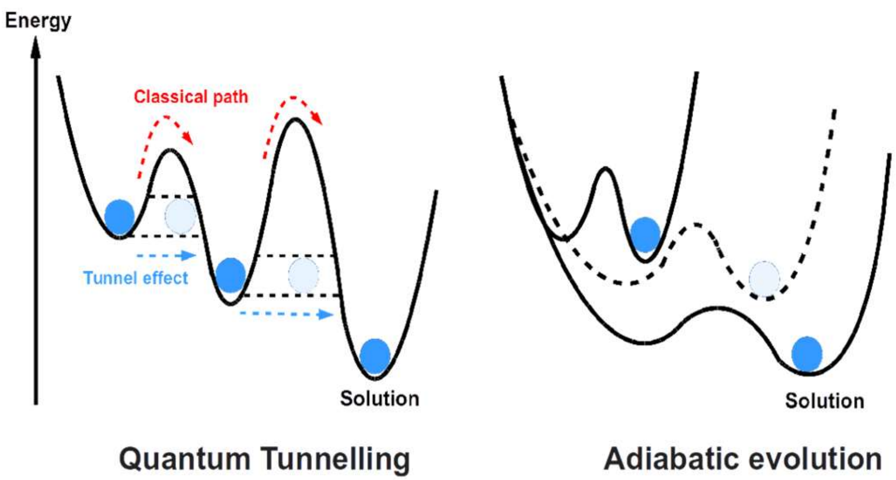
# Expected Progress of Quantum Software Engineering



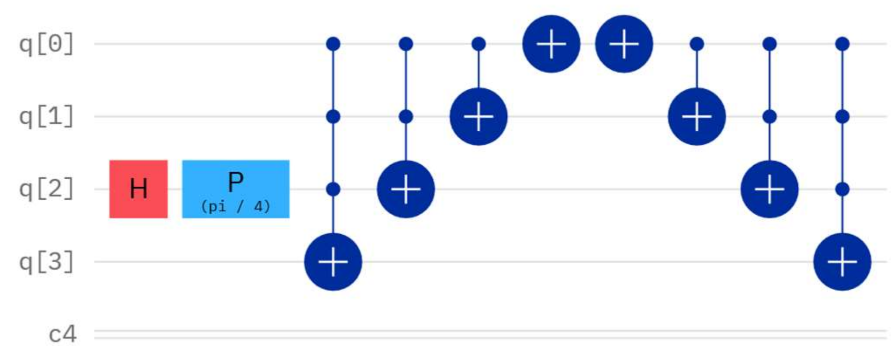
# Quantum Programming

# Quantum Programming

**Quantum annealing** (which also includes adiabatic quantum computation) is a quantum computing method used to find the optimal solution of problems involving many solutions




Universal quantum gate model is based on creating quantum structures using stable qubits and solving today's problems with **quantum circuits.** These circuits are based on the Turing machine model



# Quantum Programming

## Principal Gates

| | Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|---|
| Single Qubit | Pauli-X | x | x |
| | Hadamard | h | h |
| | Phase | p | Phaseshift |
| Multiple Qubits | Controlled-X | cx | cnot |
| | Toffoli | ccx | ccnot |
| | CPhase | cu1 | cphaseshift |
| | Swap | swap | swap |
| | Measure | measure | - |
| | Barrier | barrier | |

IBM QUANTUM COMPOSER https://quantum-computing.ibm.com/composer/files/new

# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
| --- | --- | --- |
| Pauli-X | x | x |

Composer Reference

Matrix

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Reverse

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

itself

$$A\left|0\right\rangle + B\left|1\right\rangle \;\text{—}\boxed{X}\text{—}\; B\left|0\right\rangle + A\left|1\right\rangle$$

# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| Hadamard | h | h |

Composer Reference


Matrix

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

Reverse

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

itself

$$|0\rangle \;-\!\boxed{H}\!-\; \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

$$|1\rangle \;-\!\boxed{H}\!-\; \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| Phase | p | phaseshift |

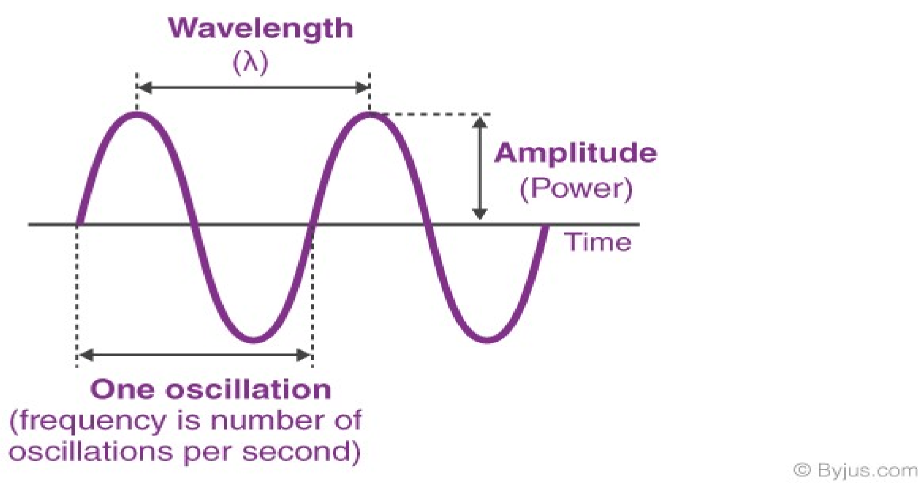
Composer Reference


Matrix

$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$

Reverse

$$P(-\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{-i\varphi} \end{bmatrix}$$



# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| Controlled-X | cx | cnot |

Composer Reference

Matrix

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

Reverse

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

# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| Toffoli | ccx | ccnot |

Composer Reference

Matrix

$$\text{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Reverse

$$\text{CCNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

itself

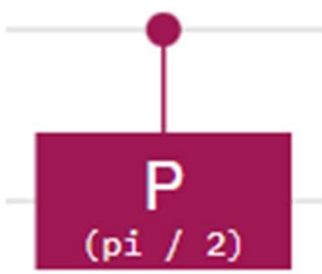
# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| CPhase | cu1 | cphaseshift |

Composer Reference

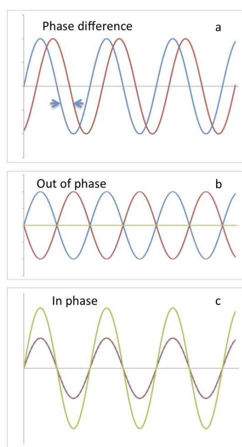
Matrix

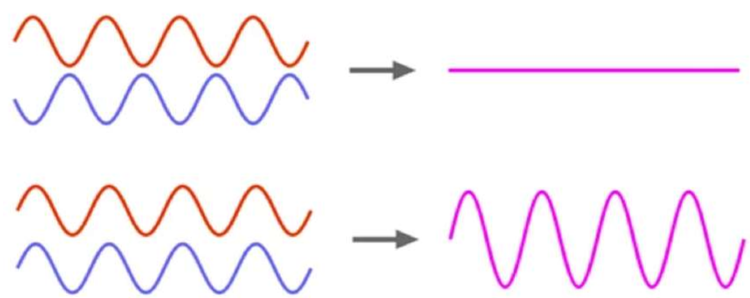
Reverse




$$\mathrm{CPHASE}(\varphi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}$$

$$CPHASE(-\varphi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\varphi} \end{bmatrix}$$



# Quantum Programming

| Name | Gate in IBM Composer (Qiskit) | Equivalent in AWS Braket |
|---|---|---|
| Swap | swap | swap |

Composer Reference

Matrix

$$\mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Reverse

$$\mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

itself

THANK YOU
VERY MUCH FOR
YOUR
ATTENTION