Quantum Simulator Report

**Background Research:**

In classical computing, programs are often written in high level languages and then compiled into an assembly language the hardware understands. The hardware then fetches, decodes, and executes these instructions sequentially and the result is stored (in a classical system) in a register or in a larger memory structure. Quantum computers obey a similar structure. To program a quantum computer, a higher-level language is often used. One of the most popular of these being Qiskit, an open software development kit from IBM that runs atop Python. This high-level code gets compiled into an assembly type language the hardware supports. This assembly type language is what this simulator runs. The language this simulator implements is a subset of the OpenQASM standard.

OpenQASM instructions allow their user to apply gates to qubits sequentially such that each gate instruction could map directly to a physical gate. According to the OpenQASM documentation, the language “possesses a dual nature as an assembly language and as a hardware description language” which, from an electrical engineer’s perspective, makes OpenQASM an interesting analog to a mix of classical software languages like C and hardware description languages like Verilog and VHDL.

**Functionality:**

**Implementation:**

This quantum simulator can be broken down into 3 major components. The simulator, the parser, and the front-end.

The simulator emulates a qubit and the affects of applying gates to qubits and takes measurements to be stored in classical registers.

The parser takes the user’s assembly code and parses it so the simulator can determine what gate to apply and with what parameters.

The front end is the web UI that the user interacts with. Thi

**Miscellaneous:**

Works Cited

AuthorLastName, FirstName. Title of the Book Being Referenced. City Name: Name of Publisher, Year. Type of Medium (e.g., Print).