```
.data
    .align 2    # Align data to word boundaries for better memory
access performance.

    # Keywords
    qubit_str: .string "qubit "         # Keyword for qubit
declaration.
    print_str: .string "print"          # Keyword for print statement.
    gate1_str: .string "qubitop1"       # Keyword for single-qubit
gate operations.
    gate2_str: .string "qubitop2"       # Keyword for two-qubit gate
operations.
    end_str: .string "end"              # Keyword to signify program
termination.

    # Error messages
    err_syntax: .string "Syntax Error\n"        # Message for general
syntax errors.
    err_var: .string "Variable not found\n"     # Message for
undefined variable access.
    err_undeclared: .string "Variable not declared\n" # Message for
undeclared variable use.

    qubitnum: .word 1      # Tracks the number of qubits in the
system.

    # Single Qubit Gates
    qubit1: .float 0.707,0,0.707,0    # Example single qubit state
(superposition).
    res: .float 0,0,0,0               # Buffer to store the result of
a gate operation.
    row1: .word 2                     # Number of rows in the first
operand.
    row2: .word 2                     # Number of columns in the first
operand (matches rows of the second operand).
    col2: .word 1                     # Number of columns in the
second operand.
```

```
    Qgate: .word 0                    # Placeholder for chosen gate
operation.

    # Common single-qubit gates.

    Hgate: .float 0.707,0,0.707,0, 0.707,0,-0.707,0   # Hadamard gate.

    Igate: .float 1,0,0,0, 0,0,1,0                  # Identity gate.

    Xgate: .float 0,0,1,0, 1,0,0,0                  # Pauli-X gate.

    Ygate: .float 0,0,0,-1, 0,1,0,0                 # Pauli-Y gate.

    Zgate: .float 1,0,0,0, 0,0,-1,0                 # Pauli-Z gate.

    Sgate: .float 1,0,0,0, 0,0,0,1                  # S gate (phase
gate).

    Tgate: .float 1,0,0,0, 0,0,0.707,0.707          # T gate (π/8
phase gate).

    SNgate: .float 0.707,0.707,0.707,-0.707, 0.707,-0.707,0.707,0.707
# Example custom gate.


    # Two Qubit Gates

    qubit2: .float 0,0,1,0       # Second qubit in a simple state.

    qubit3: .float 1,0,0,0       # Third qubit in the ground state.

    res_2: .float 0,0,0,0,0,0,0,0    # Buffer to store results of two-
qubit operations.


    tensor: .float 0,0,0,0,0,0,0,0  # Buffer for tensor product
computation.

    space: .word 0,0,0,0            # Helper array for intermediate
operations.

    row1_2: .word 4                 # Rows in the first operand (for
two-qubit operations).

    row2_2: .word 4                 # Columns in the first operand
(matches rows of the second operand).

    col2_2: .word 1                 # Columns in the second operand.

    Qgate_2: .word 1                # Placeholder for selected two-
qubit gate.


    # Common two-qubit gates.

    CNgate1: .float 1,0,0,0,0,0,0,0     # Controlled-NOT gate
configuration.
```

```
    CNgate2: .float 0,0,1,0,0,0,0,0
    CNgate3: .float 0,0,0,0,0,0,1,0
    CNgate4: .float 0,0,0,0,1,0,0,0
    SWgate1: .float 1,0,0,0,0,0,0,0     # SWAP gate.
    SWgate2: .float 0,0,0,0,1,0,0,0
    SWgate3: .float 0,0,1,0,0,0,0,0
    SWgate4: .float 0,0,0,0,0,0,1,0
    CZgate1: .float 1,0,0,0,0,0,0,0     # Controlled-Z gate.
    CZgate2: .float 0,0,1,0,0,0,0,0
    CZgate3: .float 0,0,0,0,-1,0,0,0
    CZgate4: .float 0,0,0,0,0,0,-1,0
    CYgate1: .float 1,0,0,0,0,0,0,0     # Controlled-Y gate.
    CYgate2: .float 0,0,0,0,0,0,0,-1
    CYgate3: .float 0,0,0,0,0,0,0,0
    CYgate4: .float 0,0,0,1,0,0,0,0


    # Symbol table and storage
    symbol_table: .space 1000   # Space to store mappings of variable
names to memory locations.
    value_storage: .space 2000  # Space for actual qubit state values.
    buffer: .space 100          # Input buffer for user commands.
    .align 2
    prompt: .string ">> "       # Prompt string for user input.
    .align 2
    newline: .string "\n"


.text
.globl main
main:
    # Initialize memory pointers
    la s11, value_storage  # Points to the starting location of the
value storage region.
```

```
    input_loop:
        # Print input prompt
        la a0, prompt
        li a7, 4           # Syscall for printing a string.
        ecall

        # Read user input
        la a0, buffer      # Address of the input buffer.
        li a1, 100         # Maximum input length.
        li a7, 8           # Syscall for reading a string.
        ecall

        # Check if input corresponds to a single-qubit gate operation.
        check_1Qubitgate_op:
            la t0, buffer               # Load input into temporary
register.
            la t1, gate1_str            # Load single-qubit gate
keyword.
            jal starts_with             # Check if input starts with
this keyword.
            bnez a0, check_2Qubitgate_op   # If not, move to two-qubit
gate check.

            # Handle single-qubit gate operations.
            jal ra, handle_qubitop1_stmt  # Parse and execute single-
qubit gate operations.

            beqz a0, input_loop        # Return to input loop if
operation succeeds.

            # Handle syntax errors if operation fails.
            j syntaxError

        check_2Qubitgate_op:
```

```
            la t0, buffer
            la t1, gate2_str            # Load two-qubit gate keyword.
            jal starts_with             # Check if input matches.
            bnez a0, try_next_command   # If not, try the next command
type.

            # Handle two-qubit gate operations.
            jal ra, handle_qubitop2_stmt

            beqz a0, input_loop        # Return to input loop if
operation succeeds.

            # Handle syntax errors for failed operations.
            j syntaxError

        try_next_command:
            # Check for other command types: print, qubit declaration,
or end.
            la t0, buffer
            la t1, print_str
            jal check_print_cmd
            beqz a0, handle_print_stmt

            la t0, buffer               # Example: qubit a =
[(1,0),(0,1)]
            la t1, qubit_str            # Check for qubit
declarations.
            jal starts_with
            beqz a0, handle_qubit_decl

            la t0, buffer
            la t1, end_str              # Check for "end" keyword to
terminate.
            jal check_end_cmd
```

```
            beqz a0, handle_end_cmd


        # Fall-through case: Syntax error if no command matches.
        j syntaxError


# Data related codes
.include "store.s"         # Include file for variable storage
functions.
.include "search.s"        # Include file for symbol table search
functions.


# Miscellenious codes
.include "errors.s"        # Include file for error handling.
.include "utils.s"         # Include file for utility functions.
.include "gateFunctions.s" # Include file for gate operation
functions.


# keywords related codes
.include "print.s"         # Include file for handling print
statements.
.include "qubit.s"         # Include file for qubit-specific
operations.
.include "oneQubitGate.s"  # Include file for single-qubit gate
operations.
.include "twoQubitGate.s"  # Include file for two-qubit gate
operations.
```