Program Structure:

The bytecode is organized in a list of “commands”, each starting with a command id, followed by its arguments. Each command is processed sequentially, and will be used to build the final program.

Commands:

A command consists of a command id and a variable number of arguments. The command id is a 1 byte integer, and the arguments can be of different types. The command id determines what command is being executed.

The structure of command is as follows in the bytecode:

```
<command_id> <arg1_size> <arg1> <arg2_size> <arg2> ...
command_id: 1 byte integer
arg1_size: 1 byte integer (size of the first argument)
arg1: variable size (depends on arg1_size), value of the first
argument

EX:
Let's say we have a command with id 1, which takes 2 integer
arguments and 1 string argument. The bytecode for this command
would look like this:

010201000200000861616161616161

cmd_id  arg1_size  arg1  arg2_size  arg2  arg3_size arg3
01      02         0100  02         0000  08        6161616161616
```

Arguments:

There are 5 types of arguments:

1. Integer

A 16-bit signed integer.
Important : Stored in little-endian format.

2. BigInt

A 64-bit signed integer.
Important : Stored in little-endian format.

3. String

A UTF-8 encoded string.
Important : The string is not null-terminated and stored in big-endian format.

4. Bytes

A sequence of bytes.

5. Enums

A 16-bit signed integer representing an enumeration value.
Important : Stored in little-endian format.

Here are the 2 enums used and their values :

Register:

```
0x01 -> RBX
0x02 -> R10
0x03 -> R11
0x04 -> R13
0x05 -> R14
0x06 -> R15
```

JumpCondition:

```
0x01 -> EQUAL
0x02 -> NOT_EQUAL
0x03 -> GREATER
0x04 -> GREATER_OR_EQUAL
0x05 -> LESS
0x06 -> LESS_OR_EQUAL
```