Contents

1 Introduction

This document has provided an in-depth overview of OpenLUD's bytecode native format (OBI). OpenLUD is a host of different systems for representing and executing bytecode and normal programs. This is the documentation for the OpenLUD bytecode native format, which supports multiple Operating Systems. This document explores the structure and functionality of OpenLUD's bytecode format.

2 OpenLUD's Bytecode Native Format

OpenLUD's bytecode native format, also known as OBI (OpenLUD Bytecode Intermediate), is a low-level representation of programs used by the OpenLUD system. It is designed for efficient execution and compact storage of instructions. Each OBI bytecode program is a sequence of bytes, with each statement separated by a NULL byte.

2.1 OBI Bytecode Reference

Here is a reference for the OBI bytecode instructions:

NULL (0x00): NULL is used to terminate a statement, passing control to the statement before it.

ECHO (0x40): ECHO will print out a byte as a character.

MOVE (0x41): MOVE will move a value to a register, and it will error if the register is not initialized.

EACH (0x42): EACH will print out each byte in a register if it isn't 0.

RESET (0x43): RESET will reset a register.

CLEAR (0x44): CLEAR will clear all registers.

PUT (0x45): PUT will put a VALUE into a REGISTER at a specified POSITION.

GET (0x46): GET will get a VALUE from a REGISTER at a specified POSITION and store it in another REGISTER.

INIT (0x100): INIT will initialize a register.

END (0x12): END marks the end of a bytecode program.

2.2 Bytecode Statements and Usage

In this section, we will explore how to construct bytecode statements and provide usage examples for each instruction. These examples are NOT able to be raw compiled and must be run through the compiler beforehand. See the compiler documentation for more information.

Also keep in mind that in modern OpenLUD Compilers, you will not need to use the NULL instruction to terminate statements. The compiler will provide that for you.

2.3 NULL (0x00)

The NULL instruction is used to terminate a statement, passing control to the statement before it. This means that the NULL instruction can not be used as an argument to any other instruction.

Usage Example:

```
NULL
```

2.4 ECHO (0x40)

The ECHO instruction is used to print out a byte as a character. NOTE: in modern compiler versions this instruction translates the a character to a byte. Due to the limitations of the bytecode format, this instruction will error if the character is not a valid ASCII character.

Usage Example:

```
ECHO ’A’;
```

2.5 MOVE (0x41)

The MOVE instruction is used to move a value to a register, and it will error if the register is not initialized.

Usage Example:

```
MOVE R1 ’B’;
```

2.6 EACH (0x42)

The EACH instruction is used to print out each byte in a register if it isn’t 0.

Usage Example:

```
EACH R1;
```

2.7 RESET (0x43)

The RESET instruction is used to reset a register.

Usage Example:

```
RESET R1;
```

2.8 CLEAR (0x44)

The CLEAR instruction is used to clear all registers.

Usage Example:

```
CLEAR;
```

2.9 PUT (0x45)

The PUT instruction is used to put a value into a register at a specified position.

Usage Example:

```
PUT R2 'C' 5;
```

2.10 GET (0x46)

The GET instruction is used to get a value from a register at a specified position and store it in another register.

Usage Example:

```
GET R3 2 R1;
```

2.11 INIT (0x100)

The INIT instruction is used to initialize a register.

Usage Example:

```
INIT R4;
```

2.12 END (0x12)

The END instruction marks the end of a bytecode program.

Usage Example:

```
END;
```

3 Conclusion

This document is meant to provide an in-depth overview of *OpenLUD's Bytecode Intermediate format (OBI)*. Hopefully you have a basic understanding of the OBI format, which is a good alternative for compiling native programs, which OpenLUD may not support when it comes to every Operating System, you could instead download OpenLUD's Bytecode format for your operating system and have programs that run on any device or system. This format allows for efficient execution and storage of programs in the OpenLUD system.

These are STANDARD instructions for the OpenLUD system. Any more advanced concepts like metaprogramming and bytecode optimization are meant to be covered in the compiler's documentation.