NexFUSE

NexFUSE

November 12, 2023

Contents

1 Introduction

NexFUSE(c) is a bytecode interpreter which is the successor of the OpenLUD Bytecode Intermediate format. These programs are meant to try and create a new bytecode format that is easily accessible and SAFE. This handbook is the guide to understanding NexFUSE and its commands and while this guide does not go over practical use cases, it does give examples inside of the descriptions of the commands on how they can be applied to your everyday programming. Especially if you work in computer or embedded device development, since NexFUSE contains little to no memory footprint and only allocates incremental memory, especially for lists and structures of similar nature.

NexFUSE follows an extremely, if not exact standard to OpenLUD, and is 100% organic and backwards compatible with OpenLUD.

1.1 OpenLUD & NexFUSE Speed Comparison

This is a comparison between OpenLUD and NexFUSE. OpenLUD is the bytecode intermediate format which NexFUSE is based on. NexFUSE also follows similar rules, however, these rules are used and expanded which makes NexFUSE bytecode incompatible with OpenLUD. But OpenLUD is able to create bytecode that is compatible with NexFUSE.

This is a simple speed and memory comparison between OpenLUD and NexFUSE. Do note that the same exact bytecode was run on both bytecode formats.

```
OpenLUD:

  Executed in    7.38 millis    fish           external
    usr time    4.01 millis  574.00 micros    3.44 millis
    sys time    3.44 millis    0.00 micros    3.44 millis

  valgrind (...):
    ==31954== HEAP SUMMARY:
    ==31954==     in use at exit: 124 bytes in 9 blocks
    ==31954==   total heap usage: 1,163 allocs, 1,154 frees, 606,235 bytes allocated

NexFUSE:

  Executed in    2.61 millis    fish           external
    usr time    0.34 millis  339.00 micros    0.00 millis
    sys time    2.38 millis  214.00 micros    2.17 millis

  valgrind (...):
    ==32051== HEAP SUMMARY:
    ==32051==     in use at exit: 0 bytes in 0 blocks
    ==32051==   total heap usage: 67 allocs, 67 frees, 6,605 bytes allocated
    ==32051==
    ==32051== All heap blocks were freed -- no leaks are possible
```

Using a simple 'valgrind' command, it shows that NexFUSEs memory footprint is around 25 times smaller than that of OpenLUD. Other 'time' outputs also show that NexFUSE runs 2 times faster than OpenLUD–but *HOW* you may ask? Because instead of using another programming language (D)'s garbage collector, NexFUSE instead manages its own memory, while also using an incremental-style allocation system for lists and objects.

1.2 NexFUSE Additional Features

NexFUSE contains an additional feature called **SUB** that is not a part of OpenLUD. The **SUB** command is used to define a subroutine. The following are the descriptions of the commands:

- **SUB** - defines the start of a subroutine (similar to JMP instructions)
- **ENDSUB** - defines the end of a subroutine
- **GOSUB** - jumps to a subroutine from anywhere in the program

2 NexFUSE Reference

2.1 NexFUSE Base (OpenLUD Compatible)

2.1.1 NULL (‘NNULL’)

The **NULL** command is used to end a statement call. All statements in NexFUSE AND OpenLUD are terminated with a **NULL** byte.

In all distributions of NexFUSE, **NULL** is defined as *‘00’*.

Example:

```
NULL
```

2.1.2 ECHO

The **ECHO** command is used to print out a byte as a character. NOTE that this does NOT print out a newline character after it prints the character. Since characters take up one byte, it will always print the byte as a character, there is currently no way to print out a byte as a value itself.

Example:

```
ECHO 0x41 NULL # prints ‘A’
```

2.1.3 MOVE

The **MOVE** command is used to move a byte into a register. NOTE that this updates the register pointer, and this is the only function to do so.

Example:

```
MOVE 1 0x41 NULL # moves ‘A’ into register 1
```

2.1.4 EACH

The **EACH** command loops through a register and prints each byte out if it is not **NULL/Unoccupied** (0x00).

Example:

```
MOVE 1 0x41 NULL  // move ‘A’ into register 1
MOVE 2 0x42 NULL  // move ‘B’ into register 2
MOVE 3 0x43 NULL  // move ‘C’ into register 3
MOVE 4 0x0a NULL  // move ‘\n’ into register 4
EACH 1 NULL       // prints ‘ABC\n’
```

2.1.5 PUT

The **PUT** command writes a byte to a register at a specified position. Do NOTE that the register pointer is not updated.

Example:

```
PUT 1 0x41 2 NULL # writes ‘A’ into register 1 at position 2
GET 1 2 2 NULL     # gets Register 1 at position 2 and puts it into register 2
EACH 2 NULL        # prints ‘A’
```

2.1.6 GET

The **GET** command gets a byte from a register at a specified position and puts it into another register.

Example:

```
MOVE 1 0x41 NULL  // move ‘A’ into register 1
MOVE 2 0x42 NULL  // move ‘B’ into register 2
MOVE 3 0x43 NULL  // move ‘C’ into register 3
MOVE 4 0x0a NULL  // move ‘\n’ into register 4
GET 1 2 3 NULL    // gets Register 1 at position 2 and puts it into register 3
EACH 3 NULL       // prints ‘B’
```

2.1.7 END

The **END** command is used to end a bytecode. This is used to prevent infinite loops in NexFUSE. This functionality is different than OpenLUD but END is still required in both formats.

2.1.8 RESET

The **RESET** command resets the register pointer and its values to 0.

Example:

```
RESET 1 NULL
```

2.1.9 CLEAR

The **CLEAR** command clears all registers.

Example:

```
CLEAR NULL
```

2.2 NexFUSE Additional Features

These features are not a part of the OpenLUD bytecode format. But added in NexFUSE, they do not cause any issues with memory or NexFUSEs memory footprint.

2.2.1 SUB

The **SUB** command defines the start of a subroutine. Subroutines must have an **END** as well as an **ENDSUB** command after.

Example:

```
SUB 8
  MOVE 1 0x41 NULL
  END
ENDSUB
```

2.2.2 ENDSUB

The **ENDSUB** command defines the end of a subroutine.

Example:

```
SUB 8
  MOVE 1 0x41 NULL
  END
ENDSUB
```