

# Giving nightmares to reverse engineers

---

JULES POON

GEEKCAMP 2023

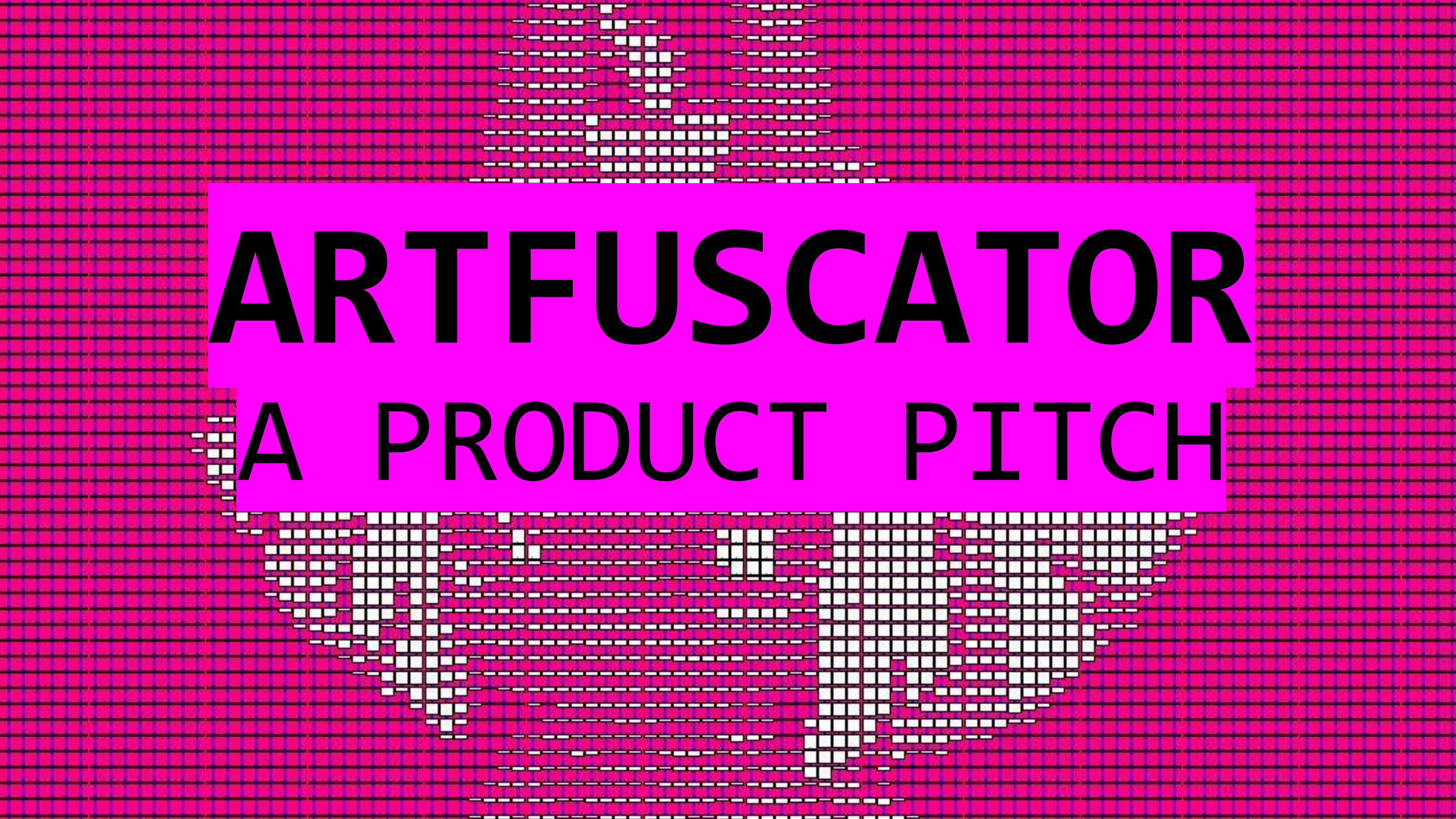
# important information about me

---

1. Student @ Somewhere

2. Likes Math and PL

3. Available  
for Intern



**ARTFUSCATOR**

**A PRODUCT PITCH**



**BACKGROUND**

# PROGRAMS: HOW THEY WORK

---

```
#if defined(RANDALL_WAS_HERE)
# define Py_UNREACHABLE() \
    Py_FatalError( \
        "If you're seeing this, the code is in what I thought was\n" \
        "an unreachable state.\n\n" \
        "I could give you advice for what to do, but honestly, why\n" \
        "should you trust me? I clearly screwed this up. I'm writing\n" \
        "a message that should never appear, yet I know it will\n" \
        "probably appear someday.\n\n" \
        "On a deep level, I know I'm not up to this task.\n" \
        "I'm so sorry.\n" \
        "https://xkcd.com/2200")
#elif defined(Py_DEBUG)
# define Py_UNREACHABLE() \
    Py_FatalError( \
        "We've reached an unreachable state. Anything is possible.\n" \
        "The limits were in our heads all along. Follow your dreams.\n" \
        "https://xkcd.com/2200")
#elif defined(__GNUC__) && (__GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__
>= 5))
# define Py_UNREACHABLE() __builtin_unreachable()
#elif defined(__clang__) || defined(__INTEL_COMPILER)
# define Py_UNREACHABLE() __builtin_unreachable()
```

<---

C PROGRAM  
(curtursy of  
cpython)



# PROGRAMS: HOW TI

--->  
Machine code  
In a PE file  
X8664



Can run on most  
intel CPUs

```
00001260 48 89 4C 24 08 48 83 EC 58 48 8B 44 24 60 48 89 H . L $ . H . . X H . D $ ` H .
00001270 44 24 38 48 8D 05 86 E1 FF FF 48 89 44 24 28 48 D $ 8 H . . . . . H . D $ ( H
00001280 8B 4C 24 28 E8 37 FB FF FF 0F B6 C0 85 C0 75 04 . L $ ( . 7 . . . . . u .
00001290 32 C0 EB 52 48 8B 44 24 28 48 8B 4C 24 38 48 2B 2 . . R H . D $ ( H . L $ 8 H +
000012A0 C8 48 8B C1 48 89 44 24 40 48 8B 54 24 40 48 8B . H . . H . D $ @ H . T $ @ H .
000012B0 4C 24 28 E8 E8 F9 FF FF 48 89 44 24 30 48 83 7C L $ ( . . . . . H . D $ 0 H . |
000012C0 24 30 00 75 04 32 C0 EB 1D 48 8B 44 24 30 8B 40 $ 0 . u . 2 . . . . . H . D $ 0 . @
000012D0 24 25 00 00 00 80 85 C0 74 04 32 C0 EB 08 B0 01 $ % . . . . . t . 2 . . . . .
000012E0 EB 04 32 C0 EB 00 48 83 C4 58 C3 CC CC CC CC CC . . 2 . . . . . H . . X . . . . .
000012F0 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001300 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001310 88 4C 24 08 48 83 EC 28 E8 92 F1 FF FF 85 C0 75 . L $ . H . . ( . . . . . u
00001320 02 EB 17 0F B6 44 24 30 85 C0 74 02 EB 0C 33 C0 . . . . . D $ 0 . . t . . . 3 .
00001330 48 8D 0D 21 72 00 00 48 87 01 48 83 C4 28 C3 CC H . . ! r . . H . . H . . ( . .
00001340 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001350 88 54 24 10 88 4C 24 08 48 83 EC 28 0F B6 05 FD . T $ . . L $ . H . . ( . . . . .
00001360 71 00 00 85 C0 74 0D 0F B6 44 24 38 85 C0 74 04 q . . . . . t . . . . D $ 8 . . t .
00001370 B0 01 EB 16 0F B6 4C 24 30 E8 BD F1 FF FF 0F B6 . . . . . L $ 0 . . . . .
00001380 4C 24 30 E8 D7 F0 FF FF B0 01 48 83 C4 28 C3 CC L $ 0 . . . . . H . . ( . .
00001390 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
000013A0 48 89 4C 24 08 48 83 EC 48 48 8B 05 B8 71 00 00 H . L $ . H . . H H . . . q . .
000013B0 48 89 44 24 30 48 83 7C 24 30 FF 75 2C 48 8B 4C H . D $ 0 H . | $ 0 . u , H . L
000013C0 24 50 E8 05 1A 00 00 85 C0 75 0C 48 8B 44 24 50 $ P . . . . . u . H . D $ P
000013D0 48 89 44 24 20 EB 09 48 C7 44 24 20 00 00 00 00 H . D $ . . . H . D $
000013E0 48 8B 44 24 20 EB 31 EB 2F 48 8B 54 24 50 48 8D H . D $ . . 1 . / H . T $ P H .
000013F0 0D 73 71 00 00 E8 C6 19 00 00 85 C0 75 0C 48 8B . s q . . . . . u . H .
00001400 44 24 50 48 89 44 24 28 EB 09 48 C7 44 24 28 00 D $ P H . D $ ( . . . H . D $ ( .
00001410 00 00 00 48 8B 44 24 28 48 83 C4 48 C3 CC CC CC . . . . . H . D $ ( H . . H . . . . .
00001420 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001430 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001440 48 89 4C 24 08 48 83 EC 38 48 8B 05 30 71 00 00 H . L $ . H . . 8 H . . 0 q . .
00001450 48 89 44 24 20 48 83 7C 24 20 FF 75 0E 48 8B 4C H . D $ . H . | $ . . u . H . L
00001460 24 40 E8 6B 19 00 00 EB 1D EB 1B 48 8B 44 24 40 $ @ . k . . . . . H . D $ @
00001470 48 89 44 24 28 48 8B 54 24 28 48 8D 0D FF 70 00 H . D $ ( H . T $ ( H . . . . p .
```

# REVERSE ENGINEERS

---

People who read machine code to understand a **binary's functionality**

- Malware researchers
- Software crackers
- Masochists

# REVERSE

People who  
understand  
**function**

- Malware
- Software
- Masoch

```
00001260 48 89 4C 24 08 48 83 EC 58 48 8B 44 24 60 48 89 H . L $ . H . . X H . D $ ` H .
00001270 44 24 38 48 8D 05 86 E1 FF FF 48 89 44 24 28 48 D $ 8 H . . . . . H . D $ ( H
00001280 8B 4C 24 28 E8 37 FB FF FF 0F B6 C0 85 C0 75 04 . L $ ( . 7 . . . . . u .
00001290 32 C0 EB 52 48 8B 44 24 28 48 8B 4C 24 38 48 2B 2 . . R H . D $ ( H . L $ 8 H +
000012A0 C8 48 8B C1 48 89 44 24 40 48 8B 54 24 40 48 8B . H . . H . D $ @ H . T $ @ H .
000012B0 4C 24 28 E8 E8 F9 FF FF 48 89 44 24 30 48 83 7C L $ ( . . . . . H . D $ 0 H . |
000012C0 24 30 00 75 04 32 C0 EB 1D 48 8B 44 24 30 8B 40 $ 0 . u . 2 . . . . . H . D $ 0 . @
000012D0 24 25 00 00 00 80 85 C0 74 04 32 C0 EB 08 B0 01 $ % . . . . . t . 2 . . . . .
000012E0 EB 04 32 C0 EB 00 48 83 C4 58 C3 CC CC CC CC CC . . 2 . . . . . H . . X . . . . .
000012F0 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001300 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001310 88 4C 24 08 48 83 EC 28 E8 92 F1 FF FF 85 C0 75 . L $ . H . . ( . . . . . u
00001320 02 EB 17 0F B6 44 24 30 85 C0 74 02 EB 0C 33 C0 . . . . . D $ 0 . . t . . . . 3 .
00001330 48 8D 0D 21 72 00 00 48 87 01 48 83 C4 28 C3 CC H . . ! r . . H . . H . . ( . .
00001340 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001350 88 54 24 10 88 4C 24 08 48 83 EC 28 0F B6 05 FD . T $ . . L $ . H . . ( . . . . .
00001360 71 00 00 85 C0 74 0D 0F B6 44 24 38 85 C0 74 04 q . . . . . t . . . . . D $ 8 . . t .
00001370 B0 01 EB 16 0F B6 4C 24 30 E8 BD F1 FF FF 0F B6 . . . . . L $ 0 . . . . .
00001380 4C 24 30 E8 D7 F0 FF FF B0 01 48 83 C4 28 C3 CC L $ 0 . . . . . H . . ( . .
00001390 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
000013A0 48 89 4C 24 08 48 83 EC 48 48 8B 05 B8 71 00 00 H . L $ . H . . H H . . . q . .
000013B0 48 89 44 24 30 48 83 7C 24 30 FF 75 2C 48 8B 4C H . D $ 0 H . | $ 0 . u , H . L
000013C0 24 50 E8 05 1A 00 00 85 C0 75 0C 48 8B 44 24 50 $ P . . . . . u . H . D $ P
000013D0 48 89 44 24 20 EB 09 48 C7 44 24 20 00 00 00 00 H . D $ . . . H . D $
000013E0 48 8B 44 24 20 EB 31 EB 2F 48 8B 54 24 50 48 8D H . D $ . . 1 . / H . T $ P H .
000013F0 0D 73 71 00 00 E8 C6 19 00 00 85 C0 75 0C 48 8B . s q . . . . . u . H .
00001400 44 24 50 48 89 44 24 28 EB 09 48 C7 44 24 28 00 D $ P H . D $ ( . . H . D $ ( .
00001410 00 00 00 48 8B 44 24 28 48 83 C4 48 C3 CC CC CC . . . H . D $ ( H . . H . . . .
00001420 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001430 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC . . . . .
00001440 48 89 4C 24 08 48 83 EC 38 48 8B 05 30 71 00 00 H . L $ . H . . 8 H . . 0 q . .
00001450 48 89 44 24 20 48 83 7C 24 20 FF 75 0E 48 8B 4C H . D $ . H . | $ . u . H . L
00001460 24 40 E8 6B 19 00 00 EB 1D EB 1B 48 8B 44 24 40 $ @ . k . . . . . H . D $ @
00001470 48 89 44 24 28 48 8B 54 24 28 48 8D 0D FF 70 00 H . D $ ( H . T $ ( H . . . p .
```

code to



Binary/machine instructions can be **DISSASSEMBLED** into a

**1. HIGH LEVEL**  
**2. HUMAN-READABLE**  
programming language known as

**ASSEMBLY**

```
.text:0000000180009F94 ; FUNCTION CHUNK AT .text:00000001800670B6 SIZE 0000008C BYTES
.text:0000000180009F94
.text:0000000180009F94      mov     [rsp+arg_0], rbx
.text:0000000180009F99      mov     [rsp+arg_8], rsi
.text:0000000180009F9E      push   rdi
.text:0000000180009F9F      sub    rsp, 20h
.text:0000000180009FA3      mov     rax, [rcx+8]
.text:0000000180009FA7      mov     rsi, rdx
.text:0000000180009FAA      mov     rbx, rcx
.text:0000000180009FAD      test   dword ptr [rax+0A8h], 10000000h
.text:0000000180009FB7      jz     loc_1800670B6
.text:0000000180009FBD      test   byte ptr [rcx+20h], 80h
.text:0000000180009FC1      jz     loc_1800670CE
.text:0000000180009FC7
.text:0000000180009FC7      loc_180009FC7:                                     ; CODE XREF: PyUnicode_AsUTF8AndSize+5D144↓j
.text:0000000180009FC7      mov     eax, [rbx+20h]
.text:0000000180009FCA      and     eax, 60h
.text:0000000180009FCD      cmp     al, 60h ; ''
.text:0000000180009FCF      lea    rax, [rbx+30h]
.text:0000000180009FD3      jnz    short loc_18000A012
.text:0000000180009FD5
.text:0000000180009FD5      loc_180009FD5:                                     ; CODE XREF: PyUnicode_AsUTF8AndSize+82↓j
.text:0000000180009FD5      test   rax, rax
.text:0000000180009FD8      jz     loc_1800670DD
.text:0000000180009FDE
.text:0000000180009FDE      loc_180009FDE:                                     ; CODE XREF: PyUnicode_AsUTF8AndSize+5D198↓j
                                                    ; PyUnicode_AsUTF8AndSize+5D1A9↓j
.text:0000000180009FDE      test   rsi, rsi
.text:0000000180009FE1      jz     short loc_180009FF4
.text:0000000180009FE3      mov     eax, [rbx+20h]
.text:0000000180009FE6      and     eax, 60h
.text:0000000180009FE9      cmp     al, 60h ; ''
.text:0000000180009FEB      jnz    short loc_18000A018
.text:0000000180009FED      mov     rax, [rbx+10h]
.text:0000000180009FF1
.text:0000000180009FF1      loc_180009FF1:                                     ; CODE XREF: PyUnicode_AsUTF8AndSize+88↓j
.text:0000000180009FF1      mov     [rsi], rax
.text:0000000180009FF4
.text:0000000180009FF4      loc_180009FF4:                                     ; CODE XREF: PyUnicode_AsUTF8AndSize+4D↑j
.text:0000000180009FF4      mov     eax, [rbx+20h]
.text:0000000180009FF7      and     eax, 60h
.text:0000000180009FFA      cmp     al, 60h ; ''
.text:0000000180009FFC      lea    rax, [rbx+30h]
.text:000000018000A000      jnz    short loc_18000A01E
```

# REVERSE ENGINEERS (RE)

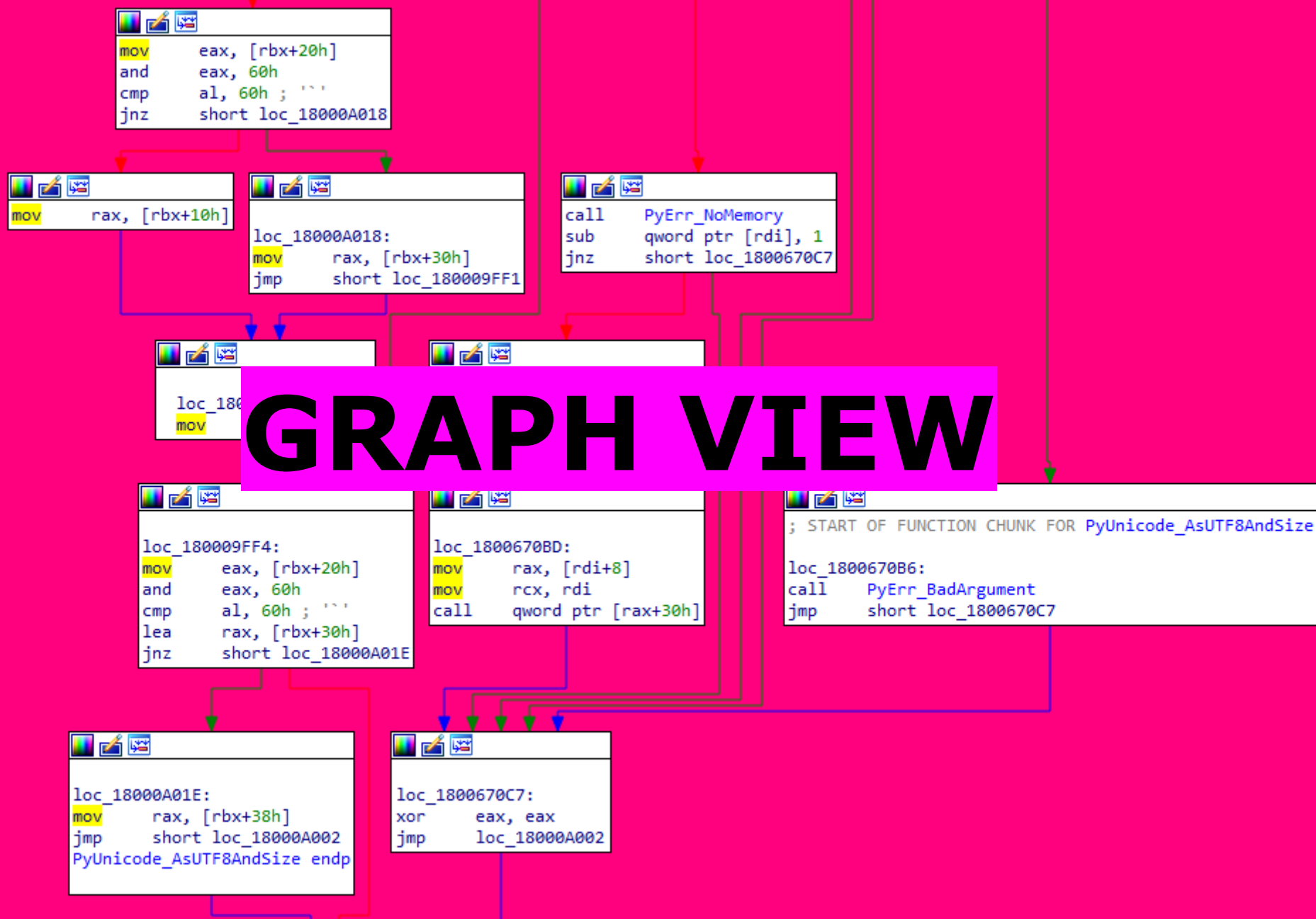
---

Tools to help them read more code at once

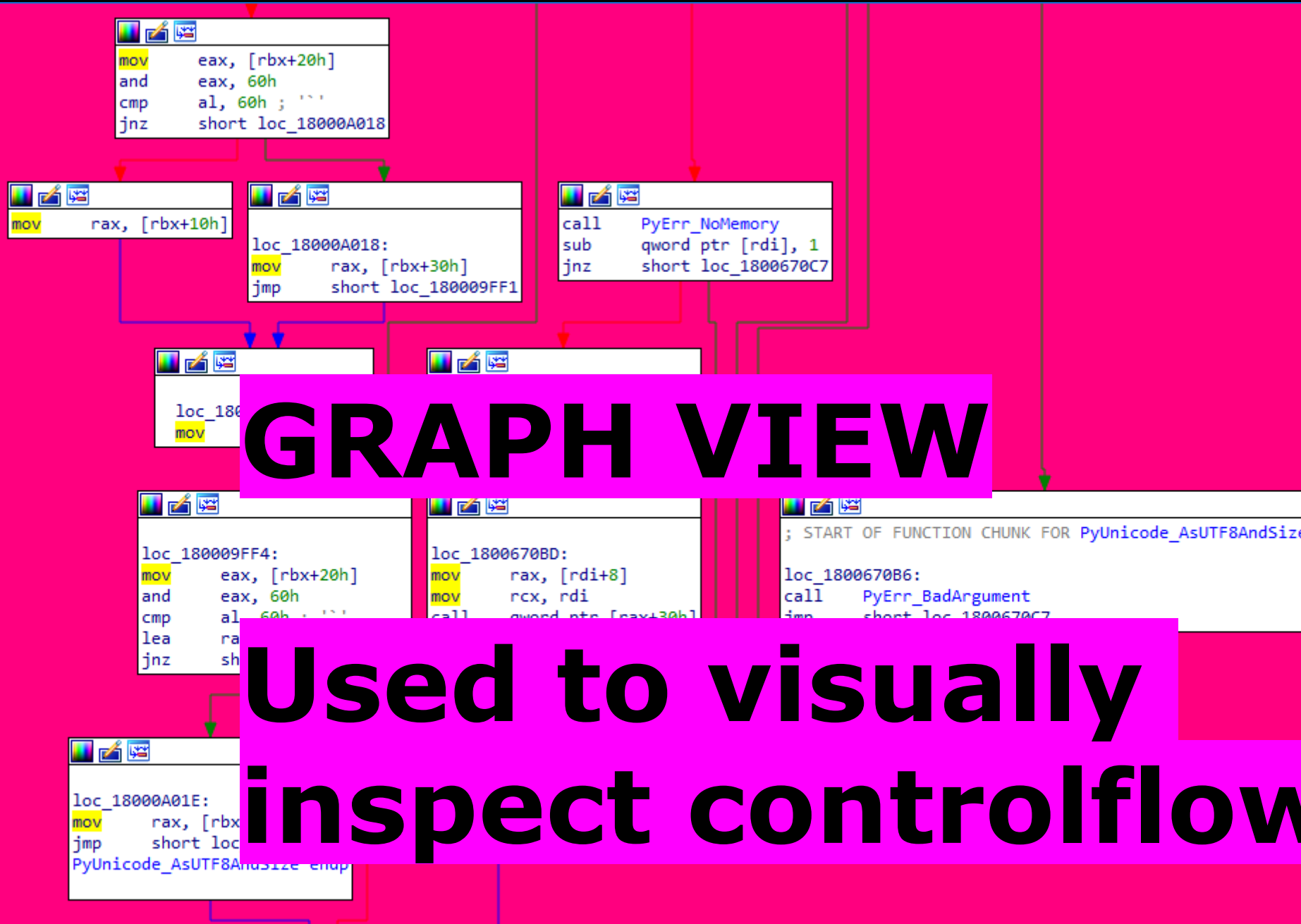
- IDA:

**Interactive DisAssembler**

The adobe photoshop for RE



# GRAPH VIEW



# GRAPH VIEW

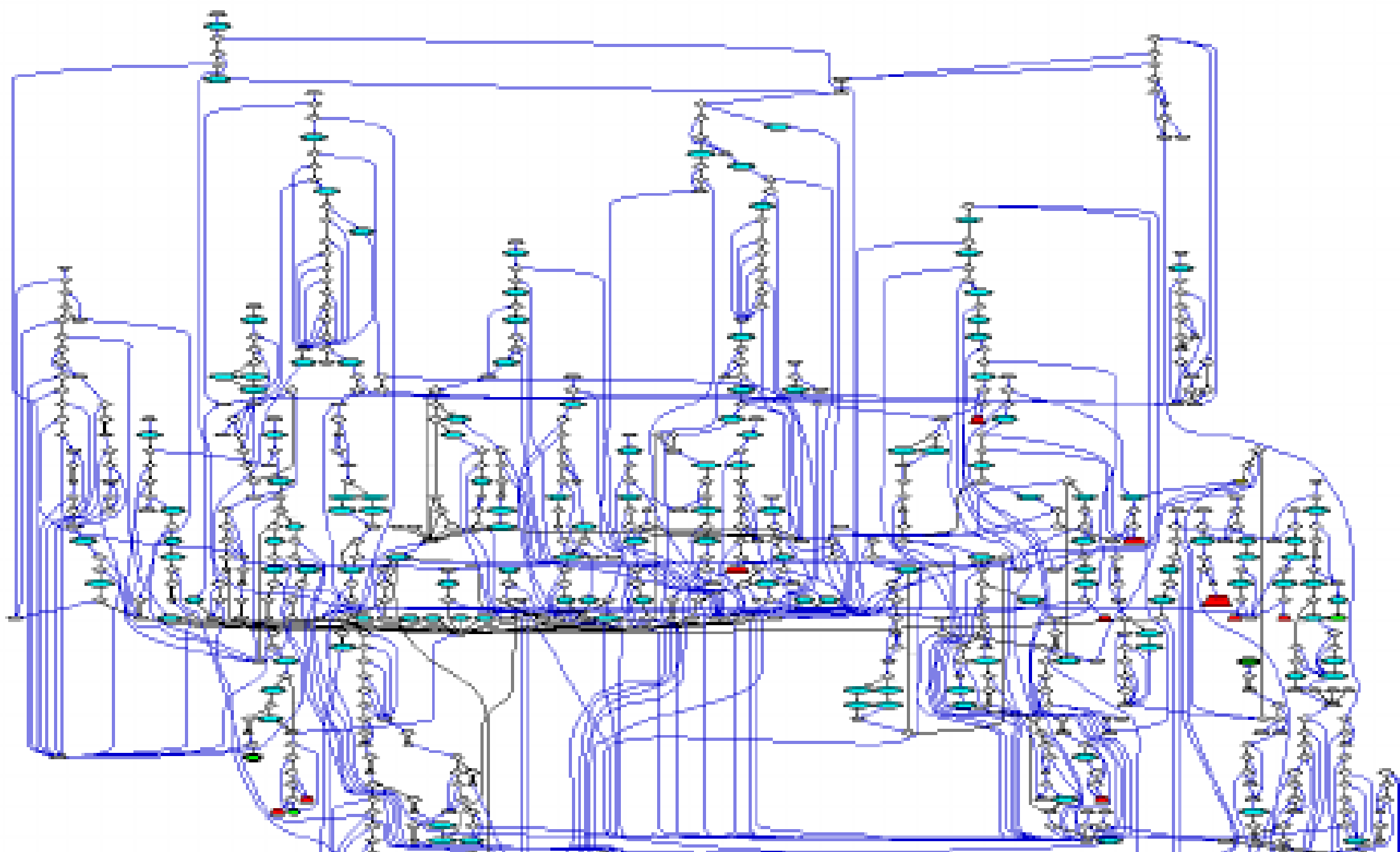
Used to visually inspect controlflow

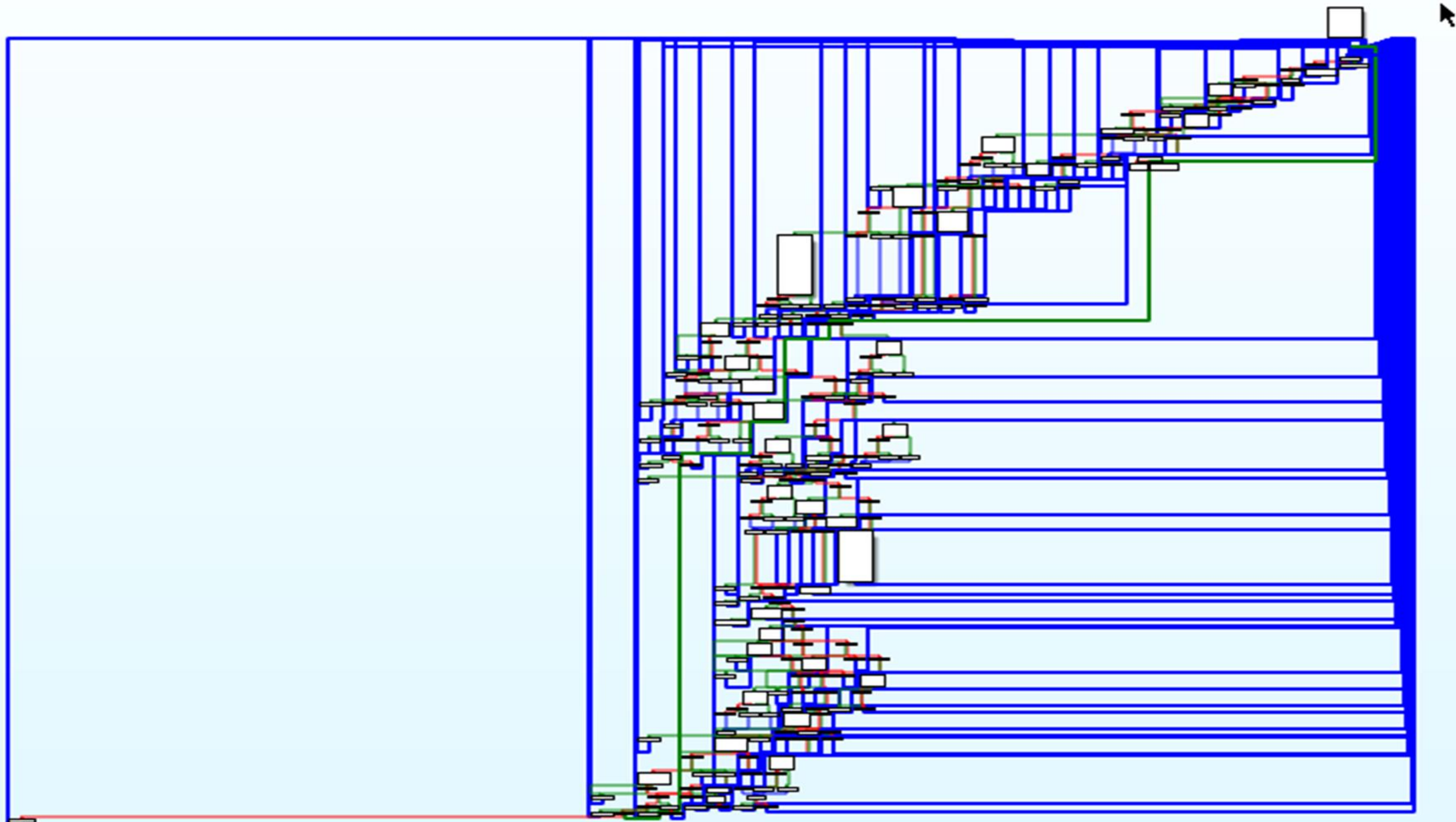
# REVERSE ENGINEERS (RE)

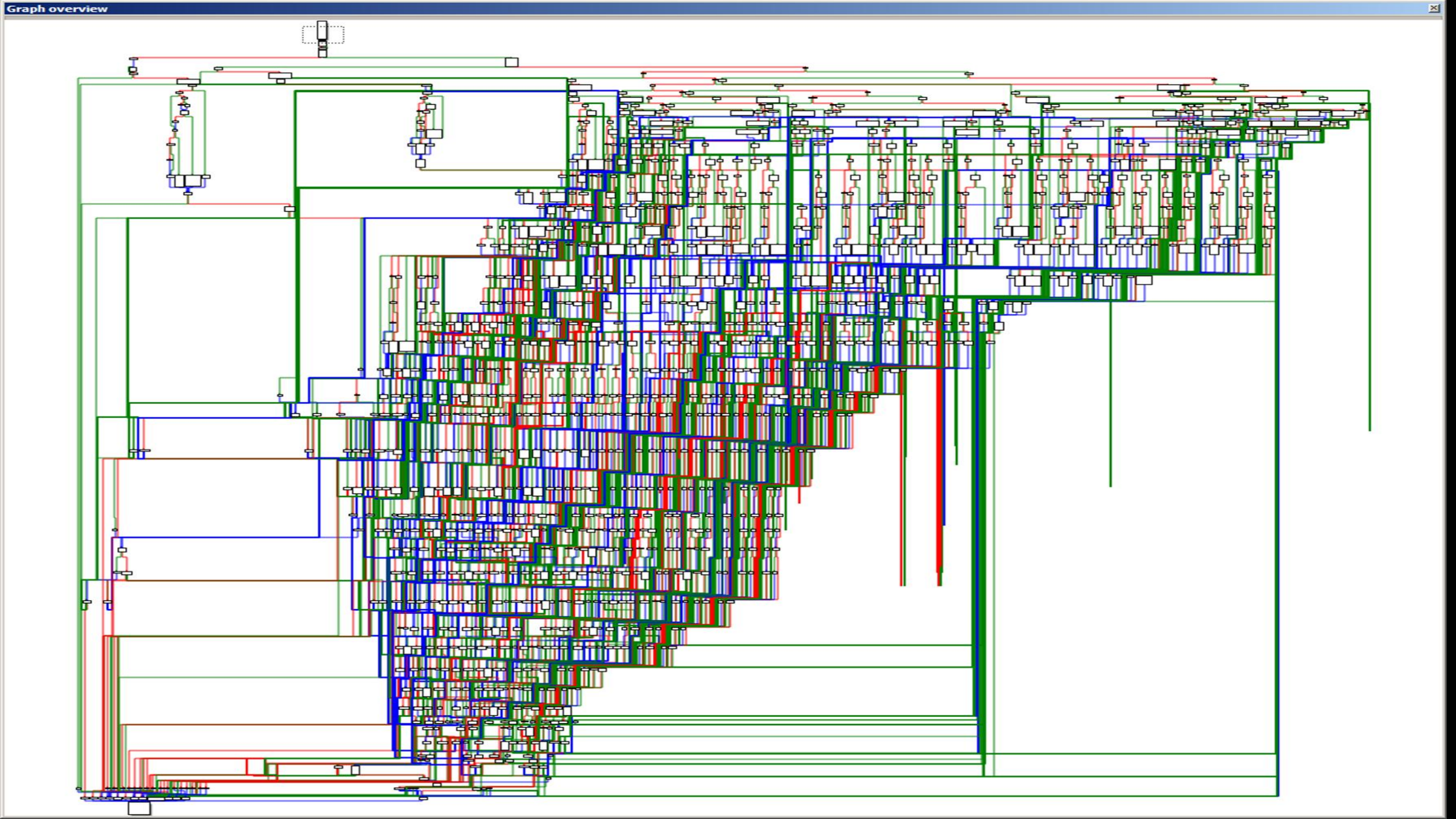
---

Reverse Engineering is a **cat and mouse game**

- Bad actors/Shitty companies keep making reversing **harder**









# REVERSE ENGINEERS (RE)


---

**Reverse Engineers are  
selectively bred to**

**achieve higher and higher  
levels of psychological  
resistance**

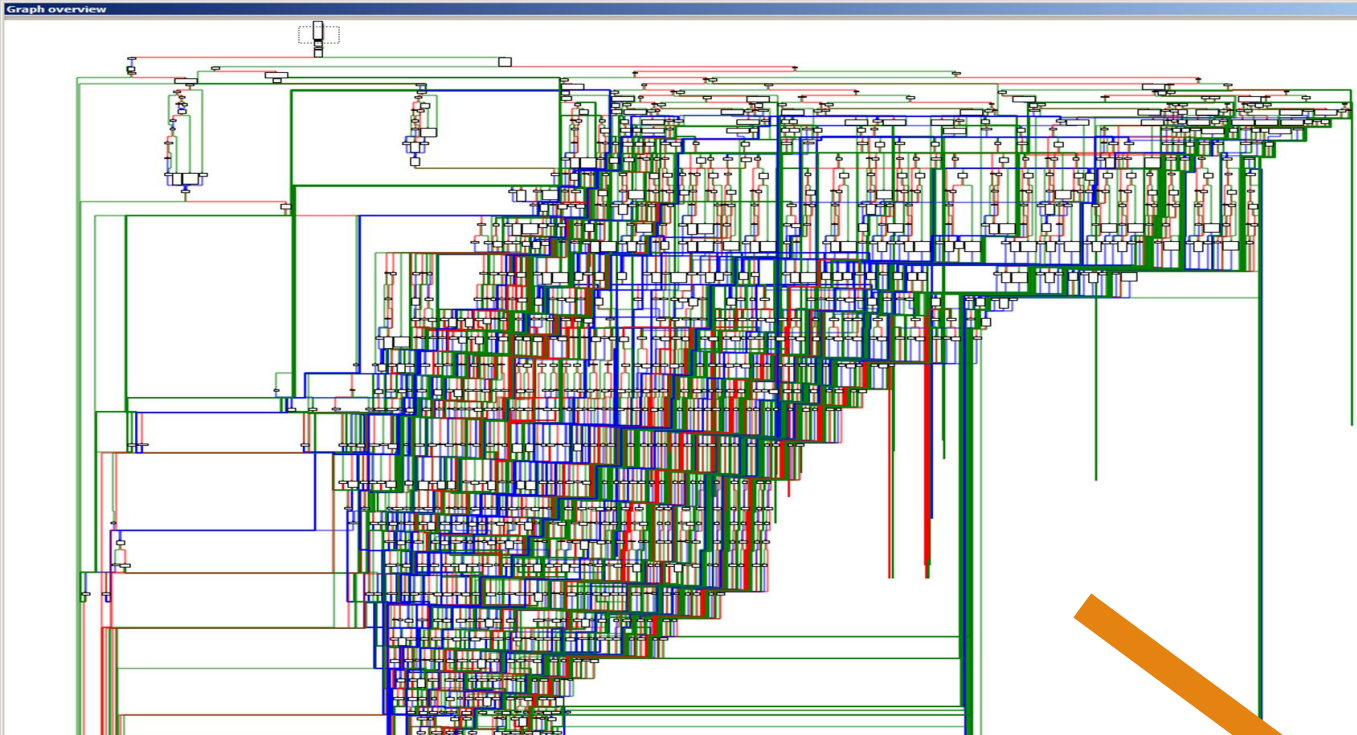


Introducing  
**ARTFUSCATOR**



Introducing  
**ARTFUSCATOR**

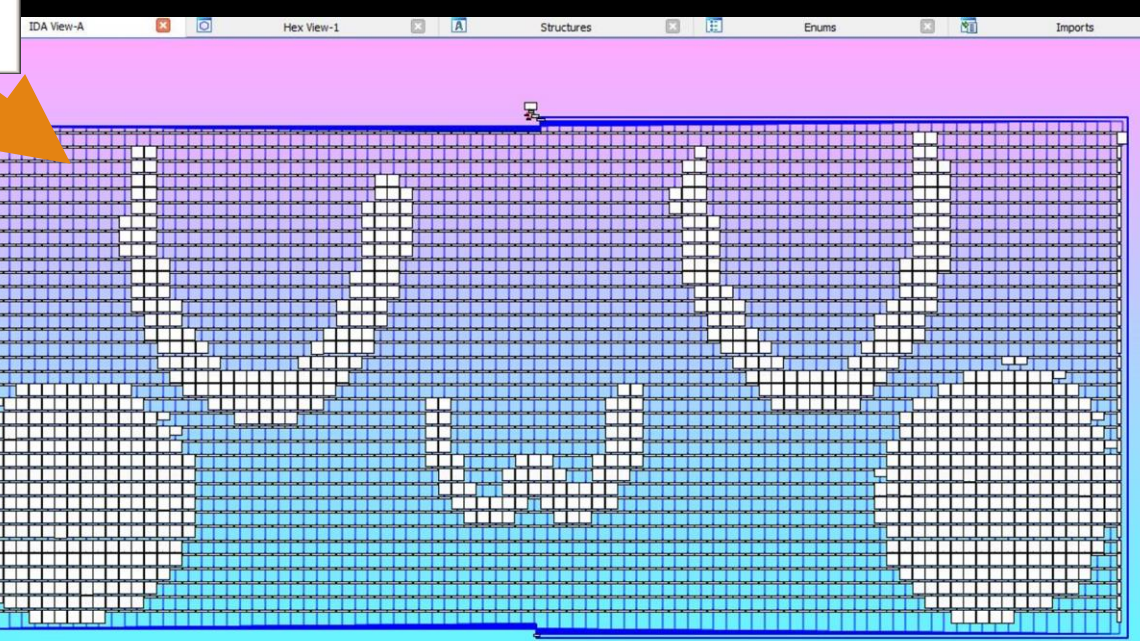
A ~~NOVEL~~ APPROACH TO  
PSYCHOLOGICAL WARFARE AGAINST  
REVERSE ENGINEERS



**A Radically New Experience**

**What's better than complex control flow**

**Than "no" controlflow at all!**



# Artfuscator

---

1. A C Compiler

2. Compiles entire program into **A SINGLE CONTROLFLOW GRAPH**  
**ART**

C hewwo.c U X

C hewwo.c > main()

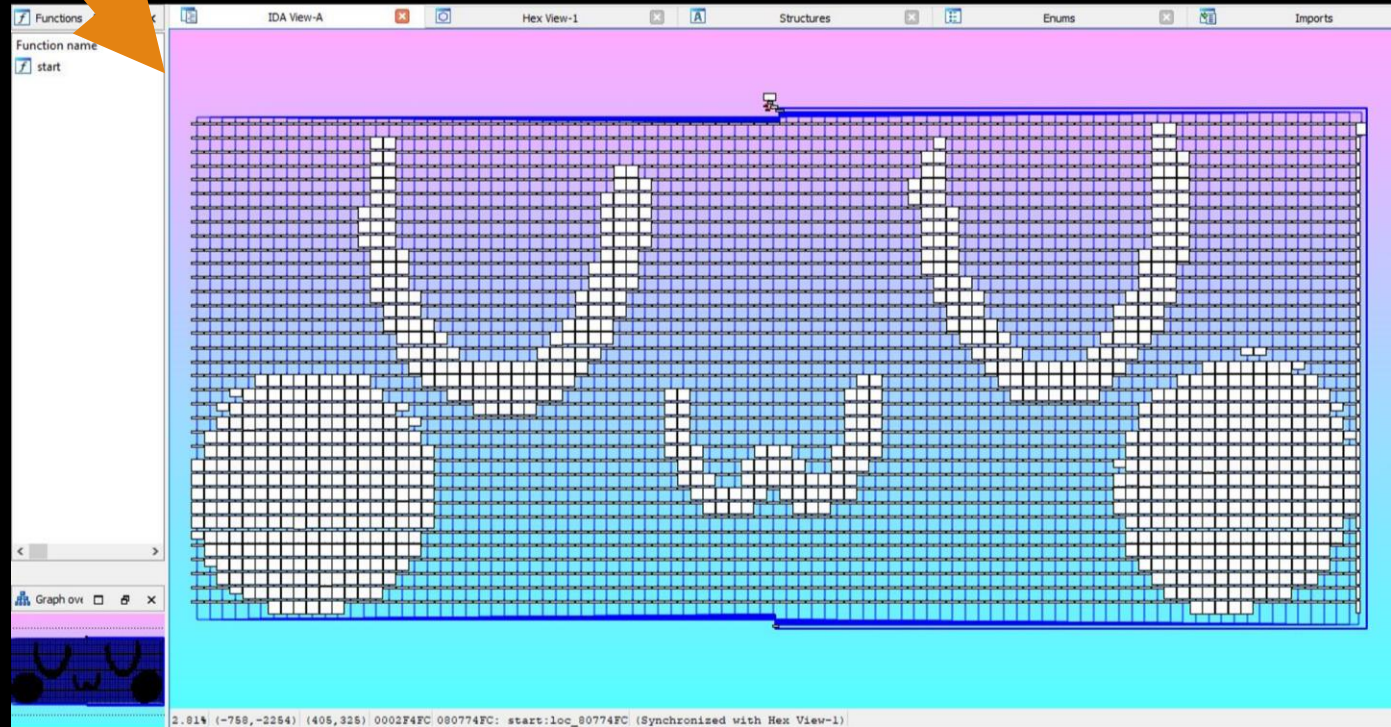
```
1 #include <stdio.h>
```

```
2
```

```
3 void main() {
```

```
4     printf("Hewwo Wowwd!! *^-^//\n");
```

```
5 }
```



# Motivation

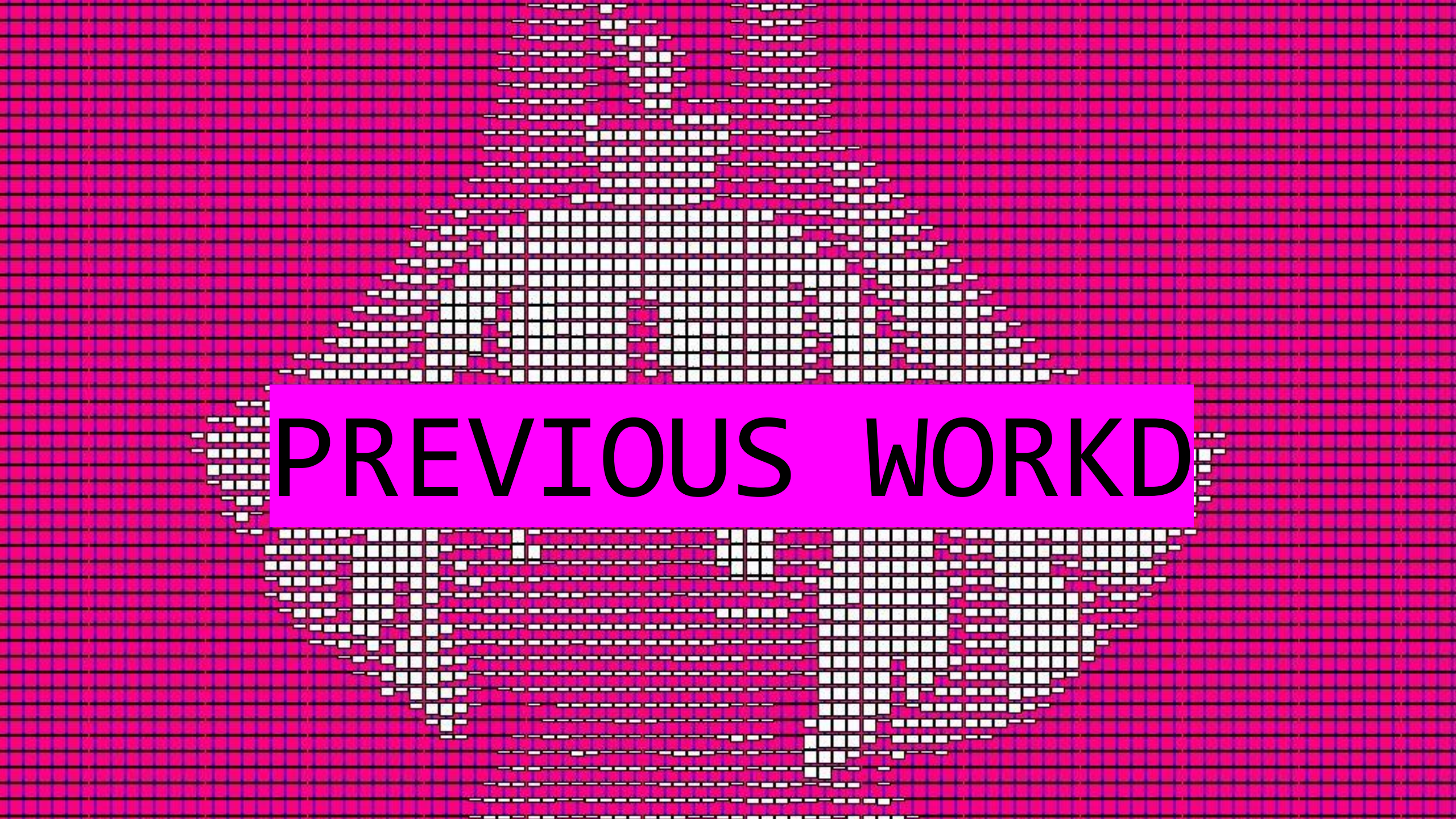
---

# Motivation

---

- covid





**PREVIOUS WORKD**

# REPSYCH by christopher domas

---



# REPSYCH by XOREAXEAXEAX

---

**1. Pioneer in Psychological Warfare in Reverse Engineering**

**2. Invented controlflow graph art**

# REPSYCH by XOREAXEAXEAX

---

**HOWEVER**

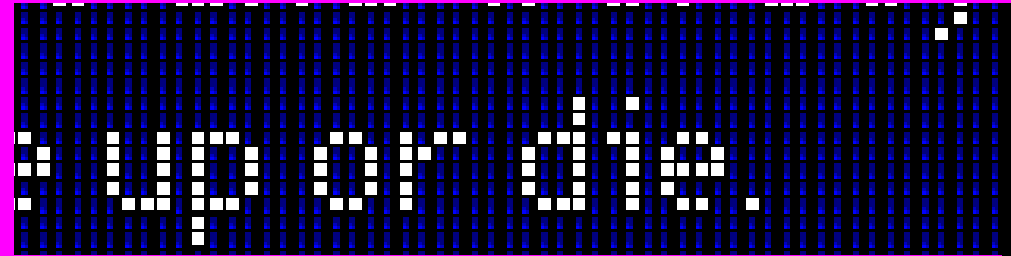
**REPSYCH HAS A FATAL FLAW**

REPSYCH by XOREAXEAXEAX

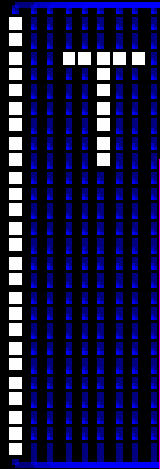
---

**THE GRAPH ART**

**HAS NO**



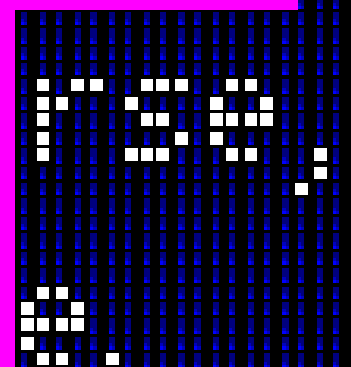
**FUNCTIONALITY**




**ARTFUSCAT**

**OR FIXES**

**THIS**





# Artfuscator Internals

# Artfuscator Internals

---

"Monumental achievement in  
compiler engineering"

(Source: me)



# Artfuscator Internals

---

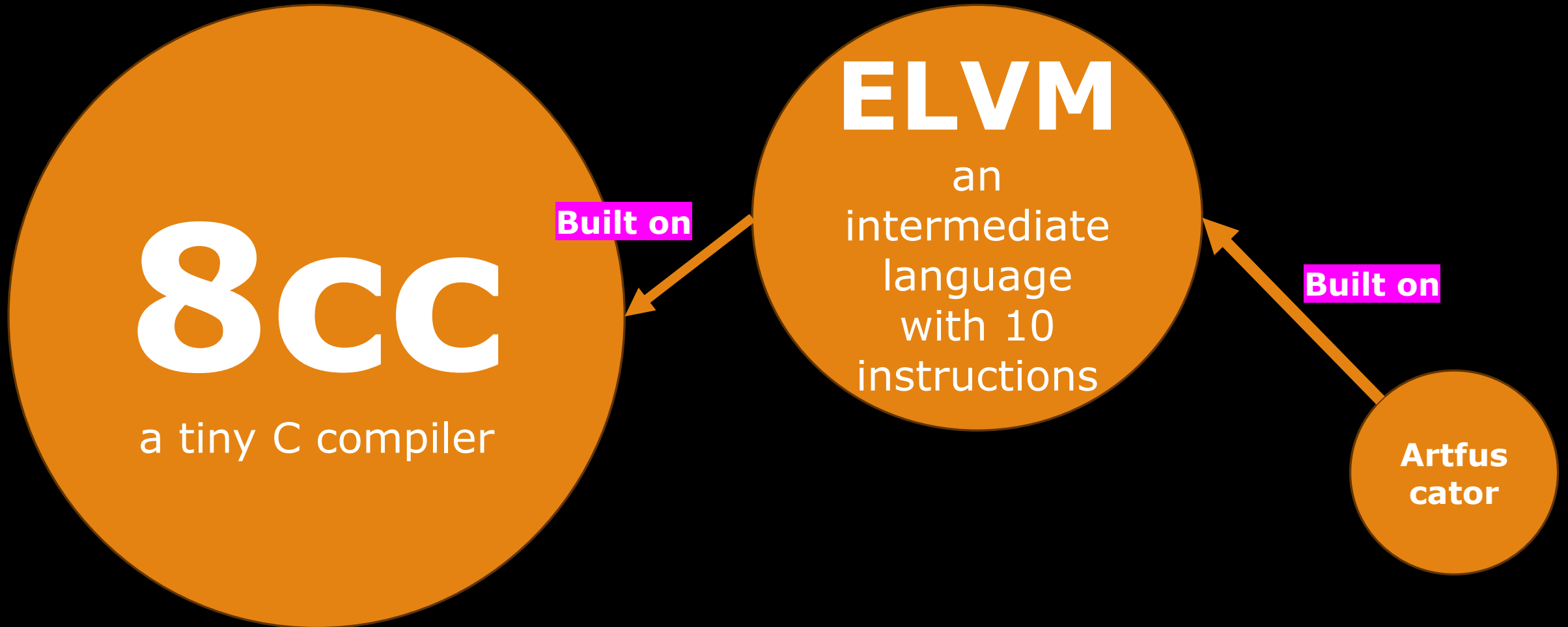
"Monumental achievement in  
compiler engineering"

(Source: me)

Artfuscator is built on the work  
of **GIANTS**

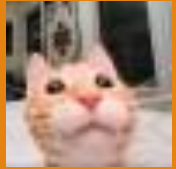
# Artfuscator Internals

---



# Artfuscator Internals

---



Rui314

Author of

Author of literally the fastest linker in the world

Correction: Rui314 wrote lld,

not ld, a linker for llvm

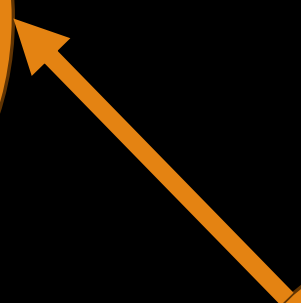
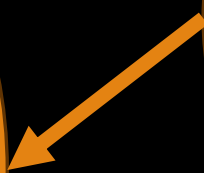
## LLVM

intermediate  
language  
with 10  
instructions

## 8cc

a tiny C compiler

Artfus  
cator



# Artfuscator Internals



**Rui314** Correction: Rui314 wrote lld, Author not ld, a linker for llvm  
Author of literally the fastest linker in the world

# 8cc

a tiny C compiler

# LLVM

intermediate



instructions

**Shinh**

Really likes brainfuck and lisp

Lots of patience

Artfus  
cator

# Artfuscator Internals



**Rui314** Correction: Rui314 wrote lld, not ld, a linker for llvm  
Author of literally the fastest linker in the world

# 8cc

a tiny C compiler

# LLVM

intermediate



instructions

**Shinh**  
Really likes brainfuck and lisp  
Lots of patience



# Credit breakdown

---

Credit Breakdown



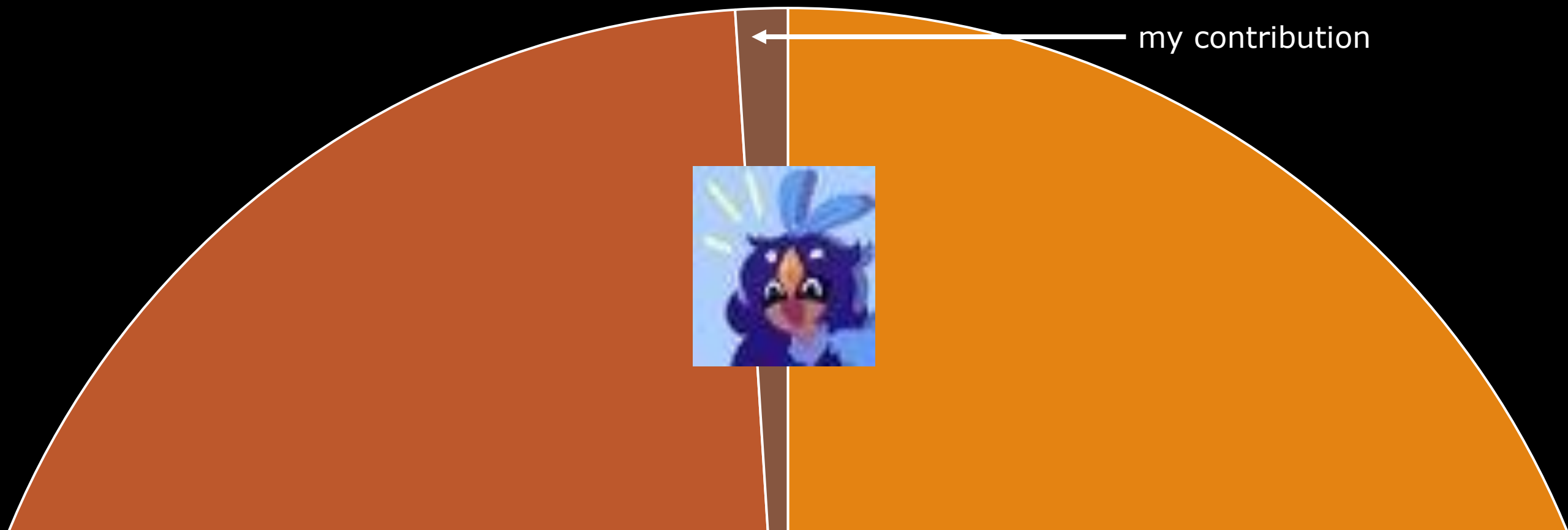
■ rui314's 8cc   ■ shinh's ELVM   ■ Jule's Contribution to the world

# Credit breakdown

---

Credit Breakdown

my contribution



# Artfuscator Internals

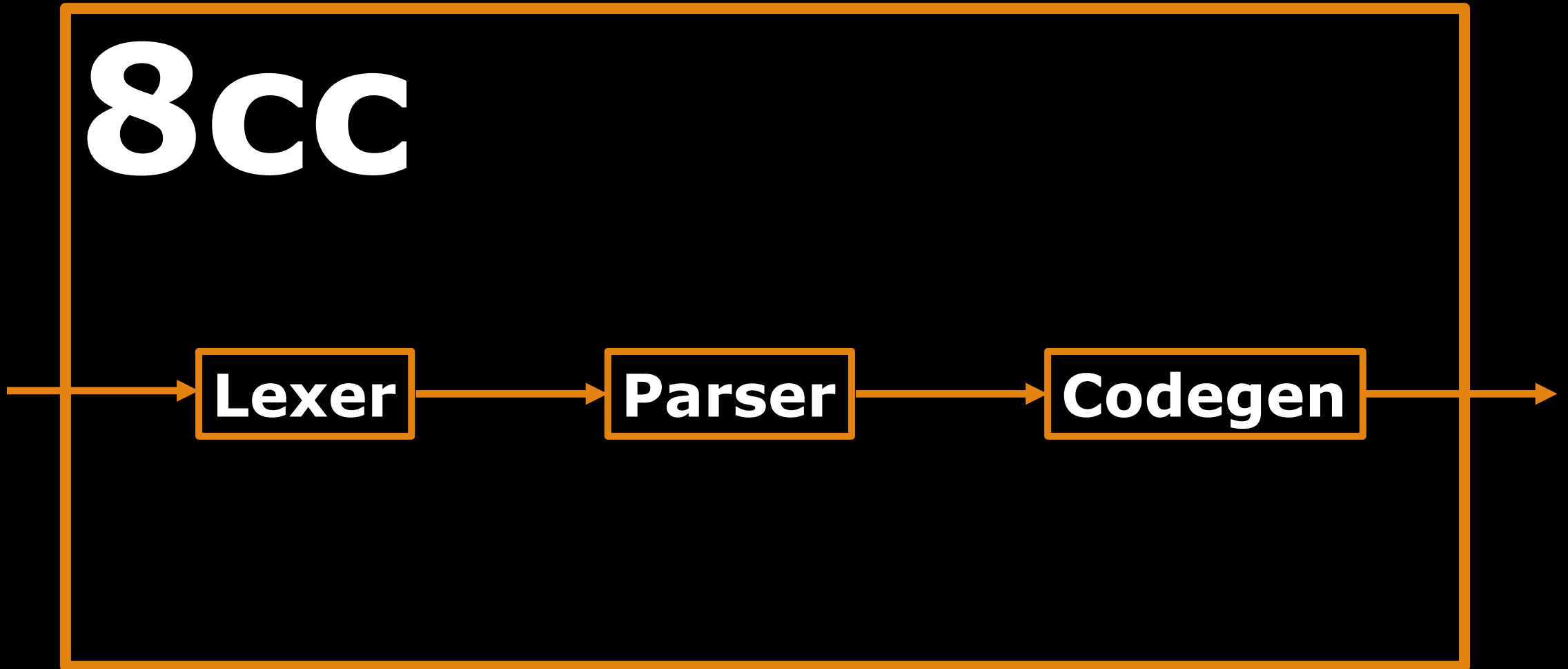
---

**8cc**

**Lexer**

**Parser**

**Codegen**





# Artfuscator Internals

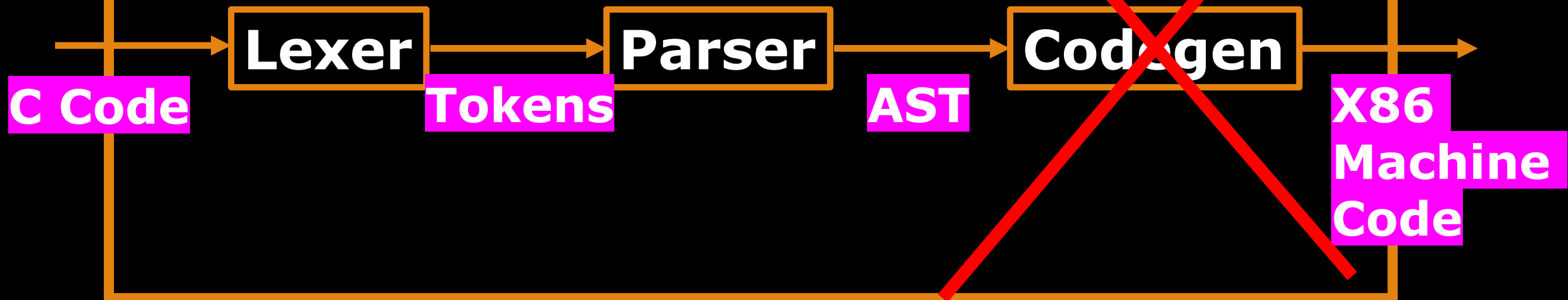
---

**8cc**



# Artfuscator Internals

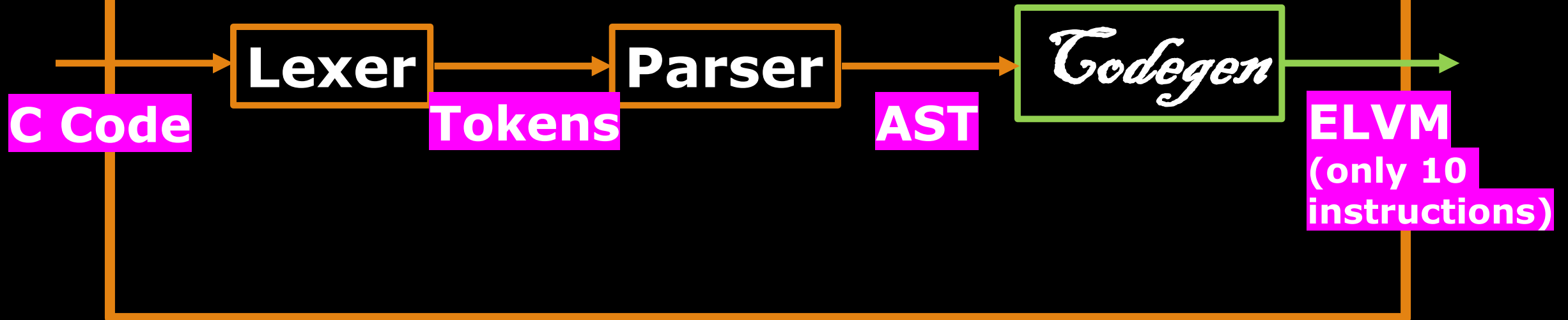
~~See~~ ELVM



# Artfuscator Internals

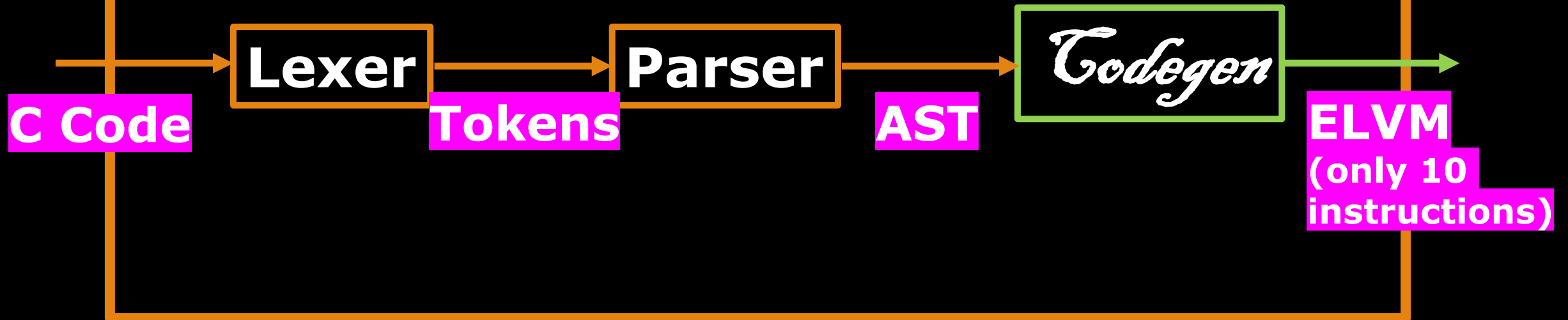
---

~~see~~ ELVM



# Artfuscator Internals

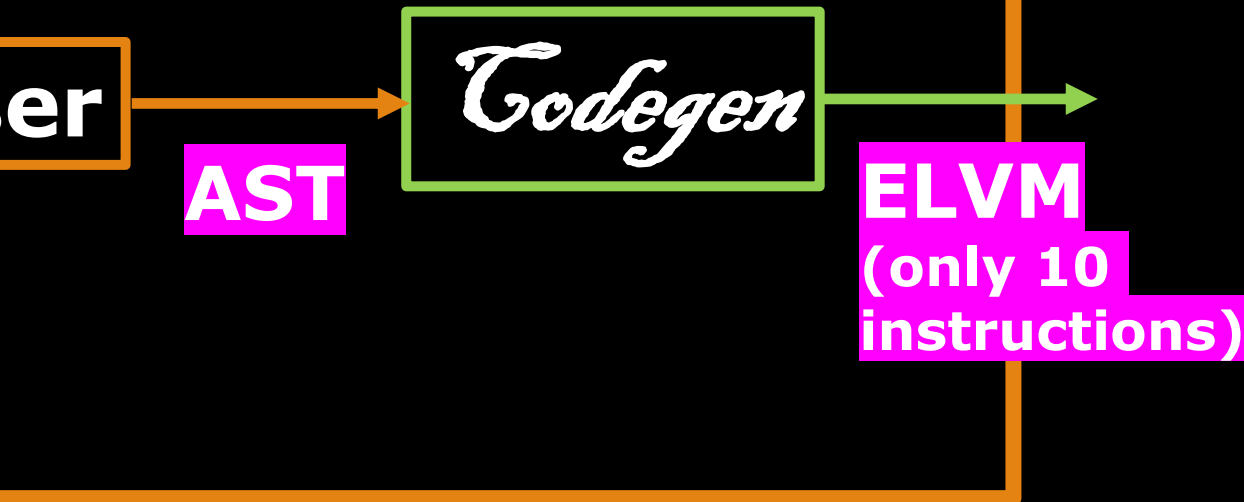
~~See ELVM Artfus~~



# Artfuscator Internals

---

# Artfuscator



# Artfuscator Internals

---

# Artfuscator



# Artfuscator Internals

---

**MY OWN  
CODEGEN**

**Job of my codegen:**

**Convert a program made up of  
10 instructions into an artful  
controlflow graph**

**mov, add, sub, load, store, setcc, jcc, putc, getc,  
exit**

# Artfuscator Internals

---

**MY OWN  
CODEGEN**

**Problem:**

**To control the graph fully we  
can't "have" controlflow**

**Can't have the program's  
controlflow interfering with  
our art**



Artfu

MY OWN  
CODEGEN

```
; void __cdecl main_main()  
main_main proc near  
  
var_18= qword ptr -18h  
var_10= qword ptr -10h  
var_1= byte ptr -1  
  
; __unwind {  
sub     rsp, 18h  
mov     [rsp+18h+var_10], rdi  
mov     rax, [rsp+18h+var_10]  
mov     [rsp+18h+var_18], rax  
mov     [rsp+18h+var_1], 1  
cmp     [rsp+18h+var_1], 0  
jz     short loc_409957
```

NO!!!

```
xor     eax, eax  
mov     edi, eax  
call    os_exit
```

```
loc_409957:  
mov     rdi, [rsp+18h+var_18]  
call    main_print_flag  
add     rsp, 18h  
retn  
; } // starts at 409930  
main_main endp
```

# Artfuscator Internals

---

## Solution?

**MY OWN  
CODEGEN**

**Do complex code transformations  
to remove controlflow entirely?**

**Spend 100s of hours crafting the  
perfect passes?**

# Artfuscator Internals Solution?

MY OWN  
CODEGEN

Do complex code transformations  
to remain **To much effort**ely?

Spend 100s of hours crafting the  
perfect passes?

# Artfuscator Internals

---

**Solution:**

**MY OWN  
CODEGEN**

**We trick disassemblers by  
converting all **jmp** into **call****

# Artfuscator Internals

---

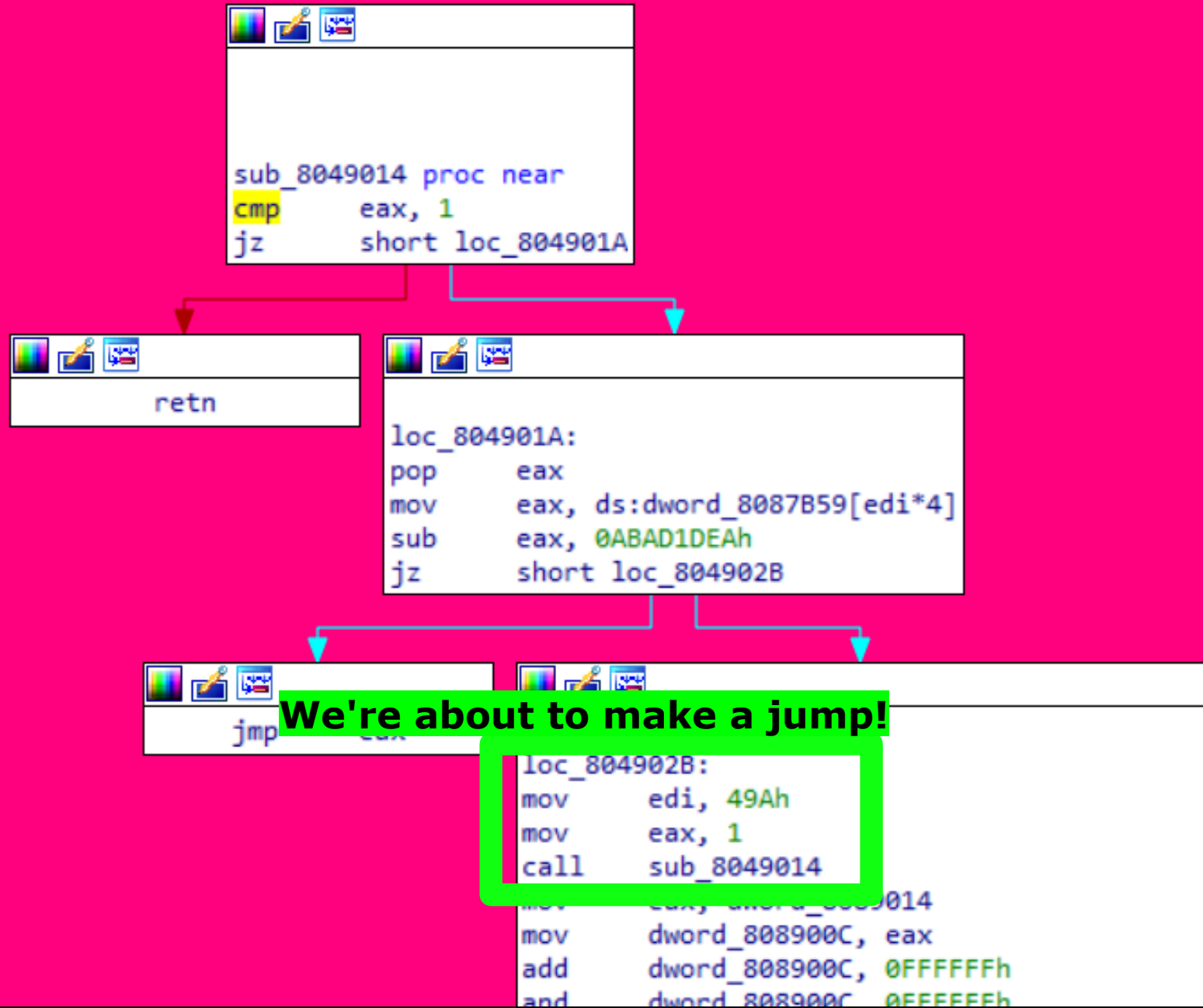
## Solution:

MY OWN  
CODEGEN

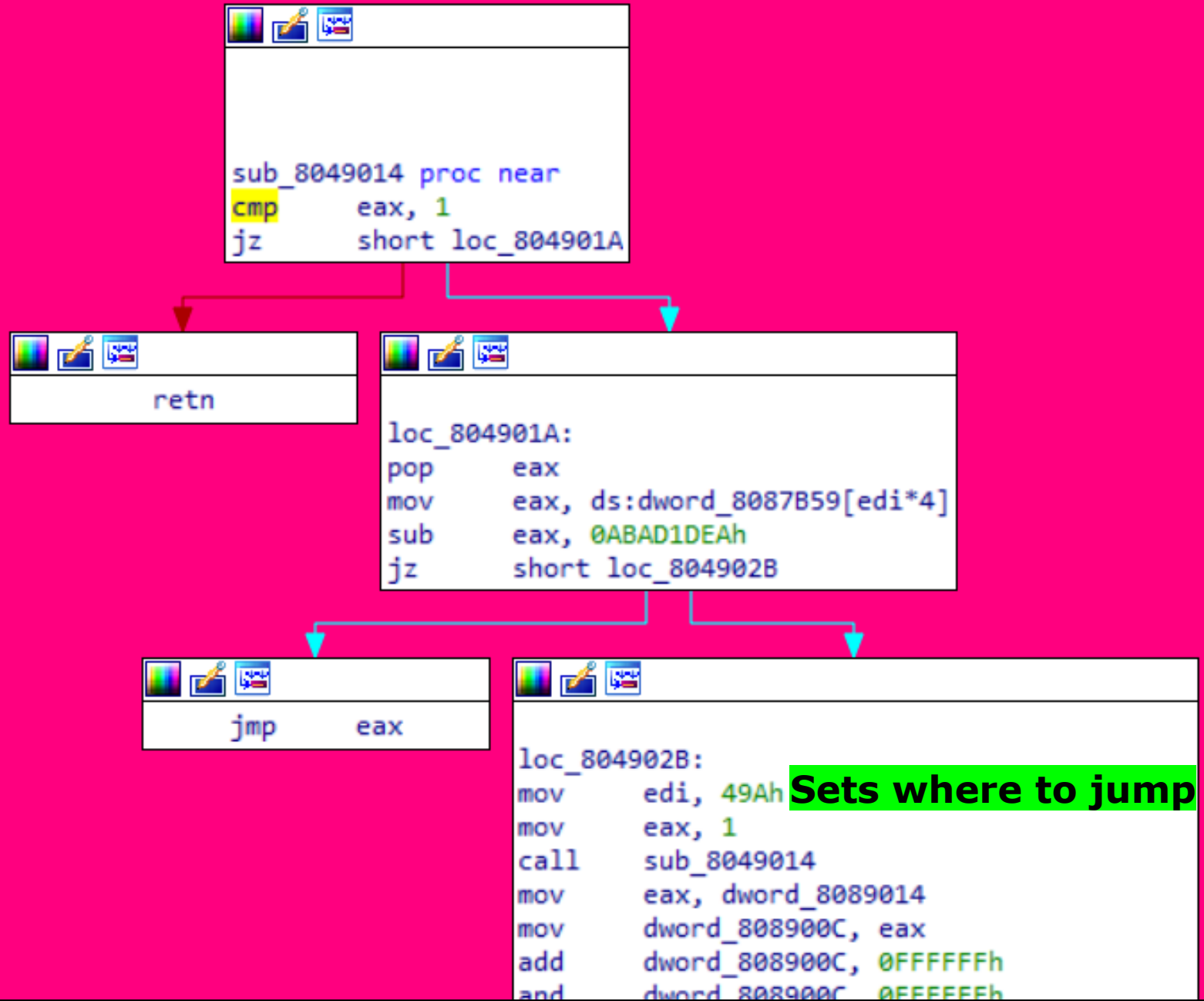
We trick disassemblers by  
converting all **jmp** into **call**

Works bcuz disassemblers  
don't split a block on a **call**

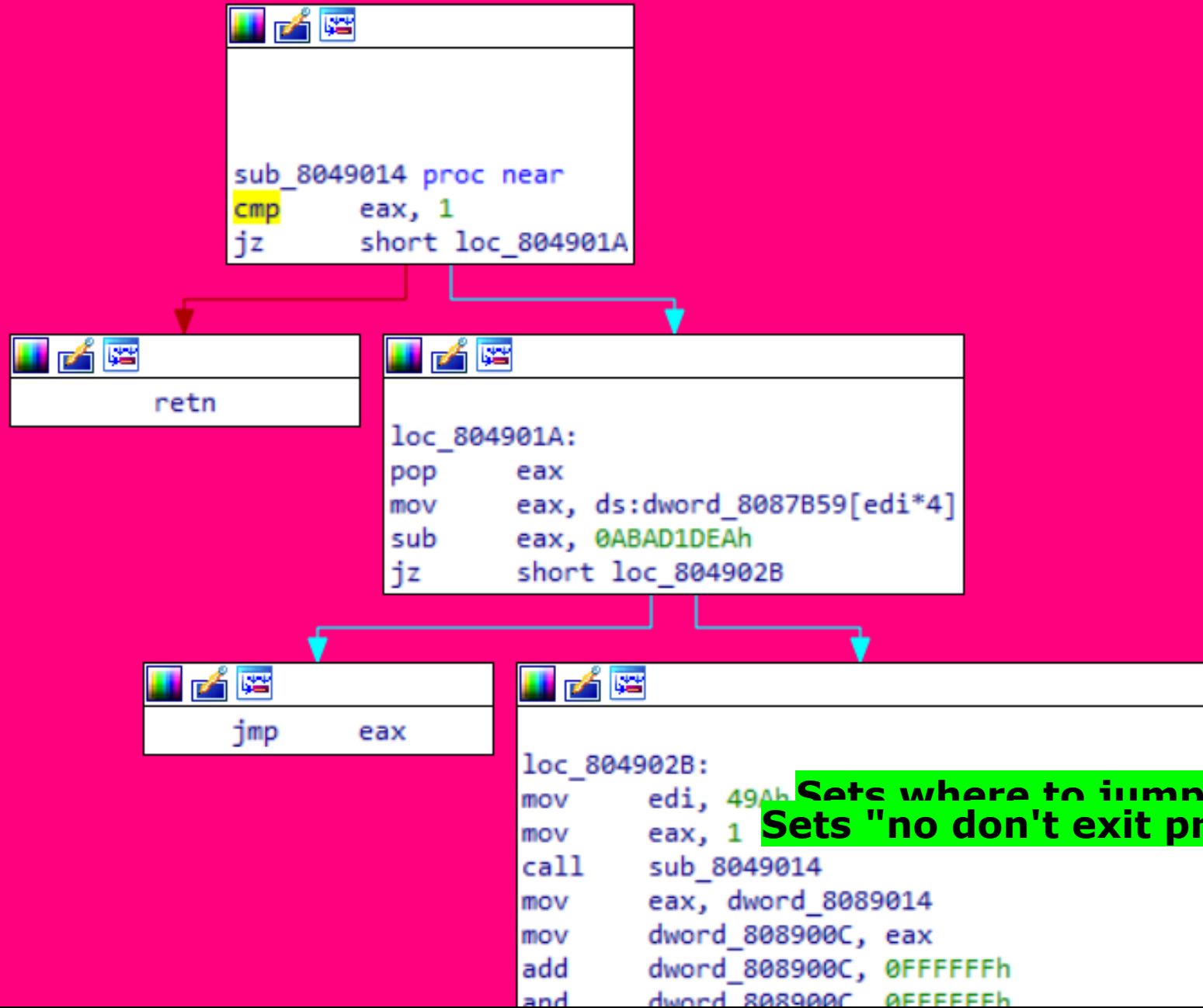
AN  
REN



AN  
REN



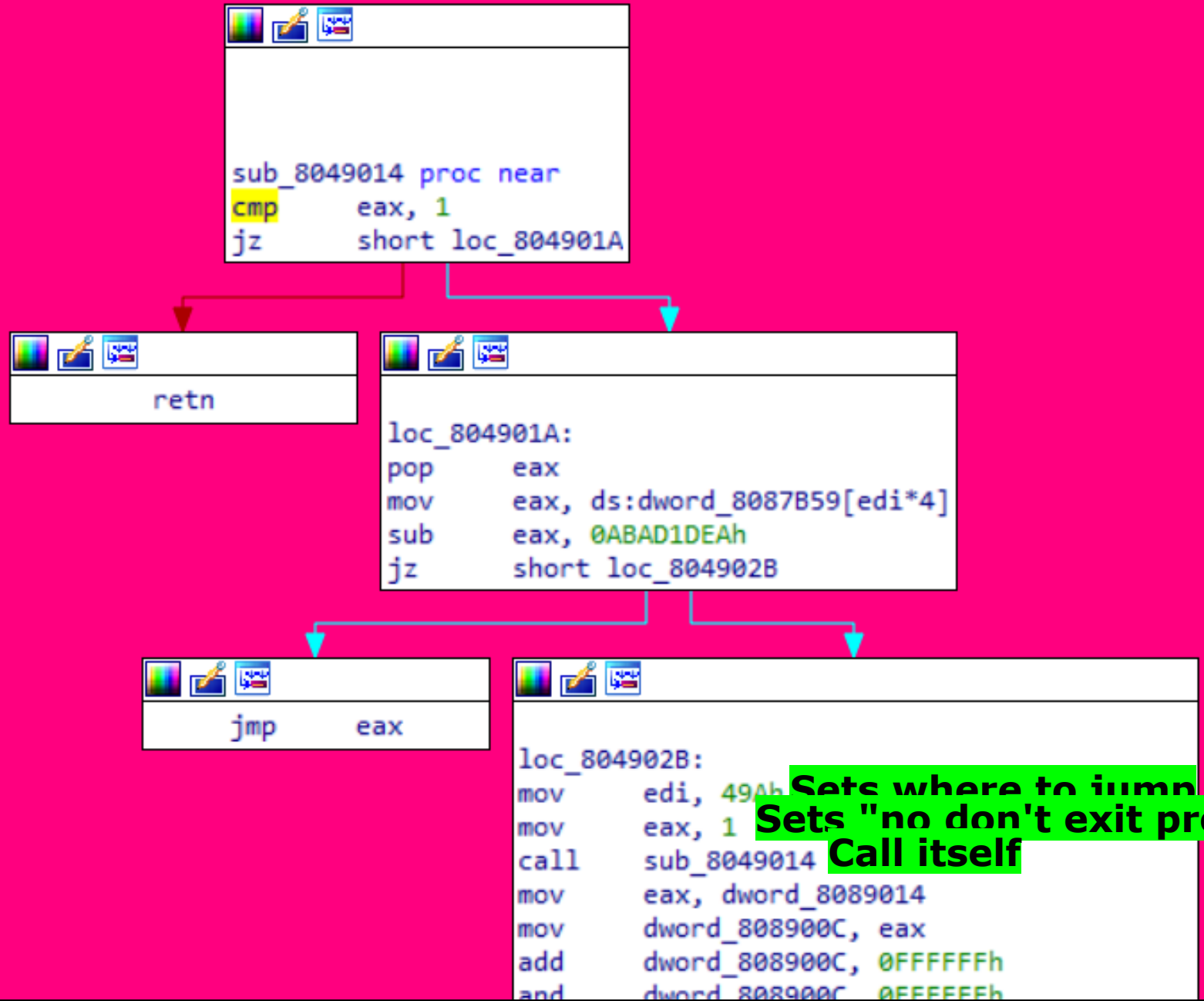
AN  
REN



Sets where to jump  
Sets "no don't exit program yet"



AN  
REN



Sets where to jump  
Sets "no don't exit program yet"  
Call itself

AN  
REN

```
sub_8049014 proc near  
cmp    eax, 1  
jz     short loc_804901A
```

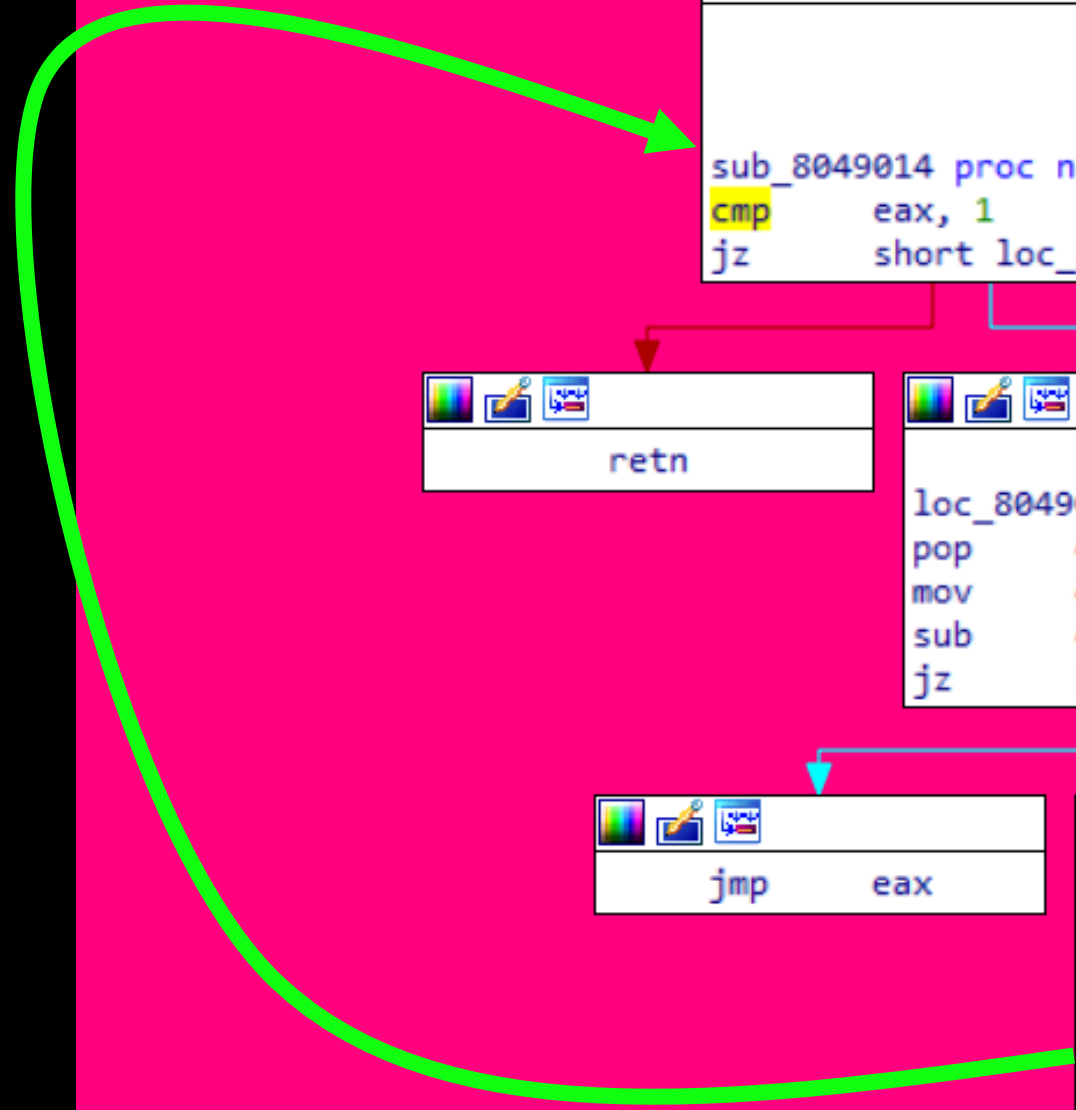
```
retn
```

```
loc_804901A:  
pop    eax  
mov    eax, ds:dword_8087B59[edi*4]  
sub    eax, 0ABAD1DEAh  
jz     short loc_804902B
```

```
jmp    eax
```

```
loc_804902B:  
mov    edi, 49Ah  
mov    eax, 1  
call   sub_8049014  
mov    eax, dword_8089014  
mov    dword_808900C, eax  
add    dword_808900C, 0FFFFFFh  
and    dword_808900C, 0FFFFFFh
```

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



AN  
REN

```
sub_8049014 proc near  
cmp    eax, 1  
jz     short loc_804901A
```

```
retn
```

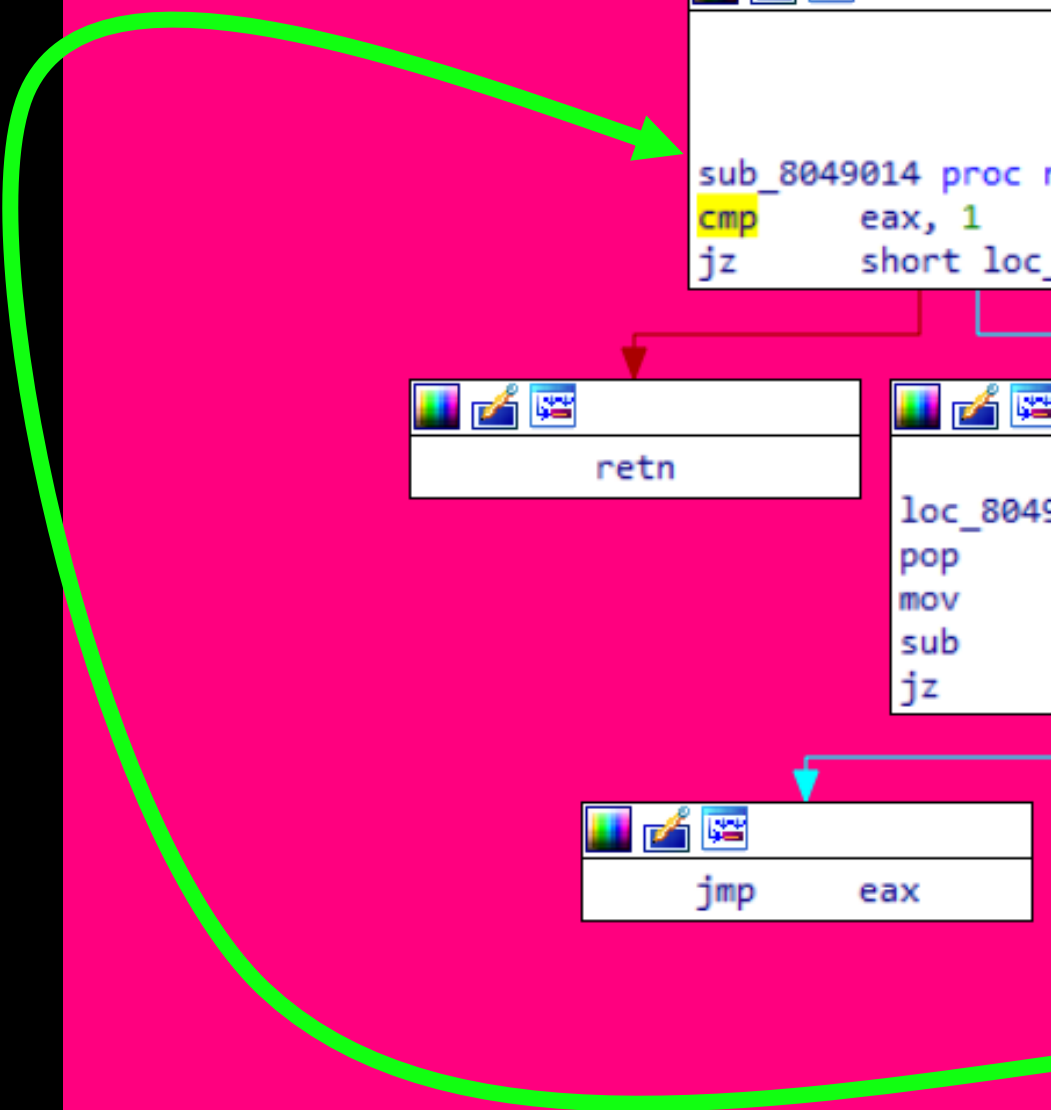
```
loc_804901A:  
pop    eax  
mov    eax, ds:dword_8087B59[edi*4]  
sub    eax, 0ABAD1DEAh  
jz     short loc_804902B
```

```
jmp    eax
```

```
loc_804902B:  
mov    edi, 49Ah  
mov    eax, 1  
call   sub_8049014  
mov    eax, dword_8089014  
mov    dword_808900C, eax  
add    dword_808900C, 0FFFFFFh  
and    dword_808900C, 0FFFFFFh
```

Removes useless return address from stack

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



AN  
EN

```
sub_8049014 proc near  
cmp    eax, 1  
jz     short loc_804901A
```

```
retn
```

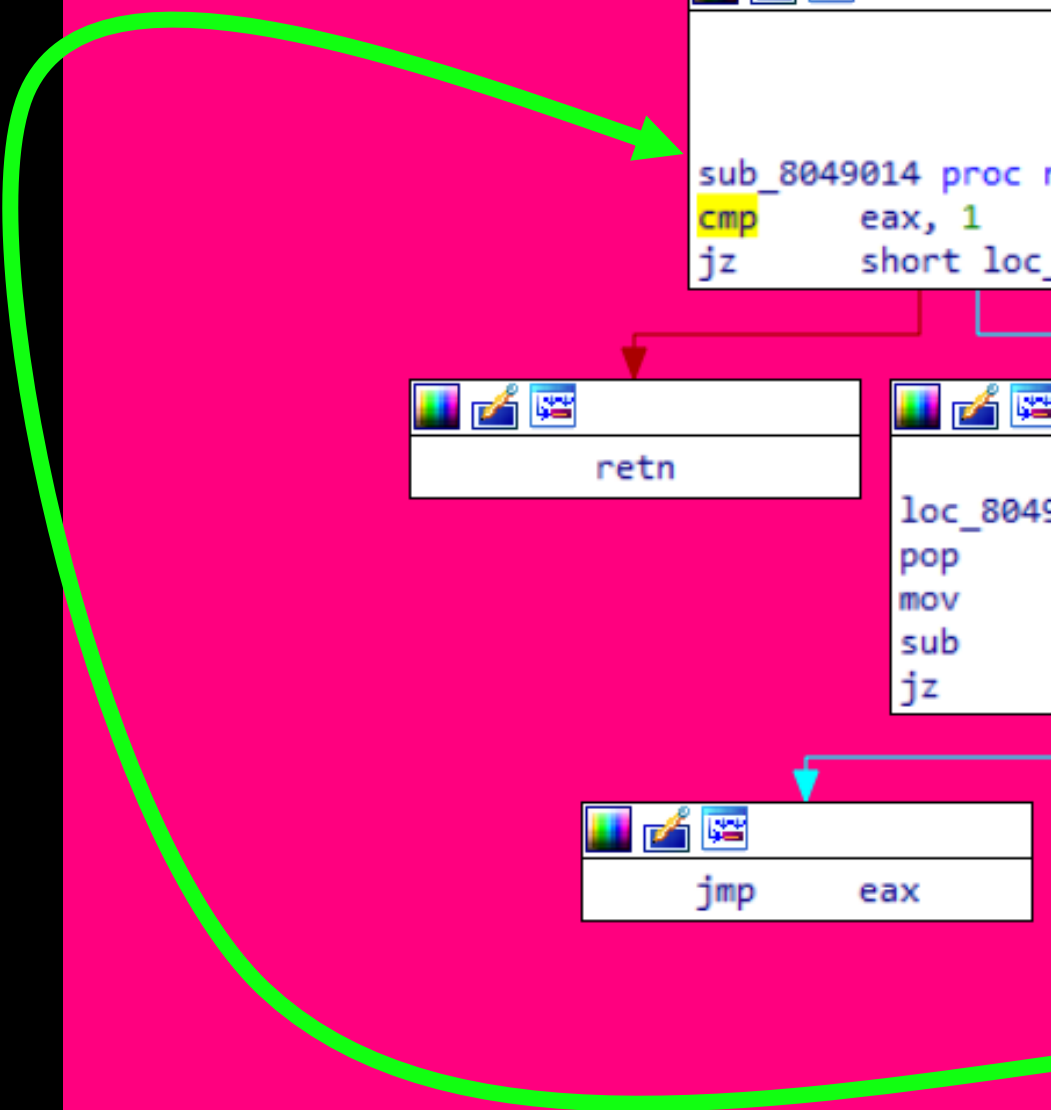
```
loc_804901A:  
pop    eax  
mov    eax, ds:dword_8087B59[edi*4]  
sub    eax, 0ABAD1DEAh  
jz     short loc_804902B
```

```
jmp    eax
```

```
loc_804902B:  
mov    edi, 49Ah  
mov    eax, 1  
call   sub_8049014  
mov    eax, dword_8089014  
mov    dword_808900C, eax  
add    dword_808900C, 0FFFFFFh  
and    dword_808900C, 0FFFFFFh
```

Removes useless return address from stack  
Lookup address to jump to

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



AN  
EN

```
sub_8049014 proc near  
cmp    eax, 1  
jz     short loc_804901A
```

```
retn
```

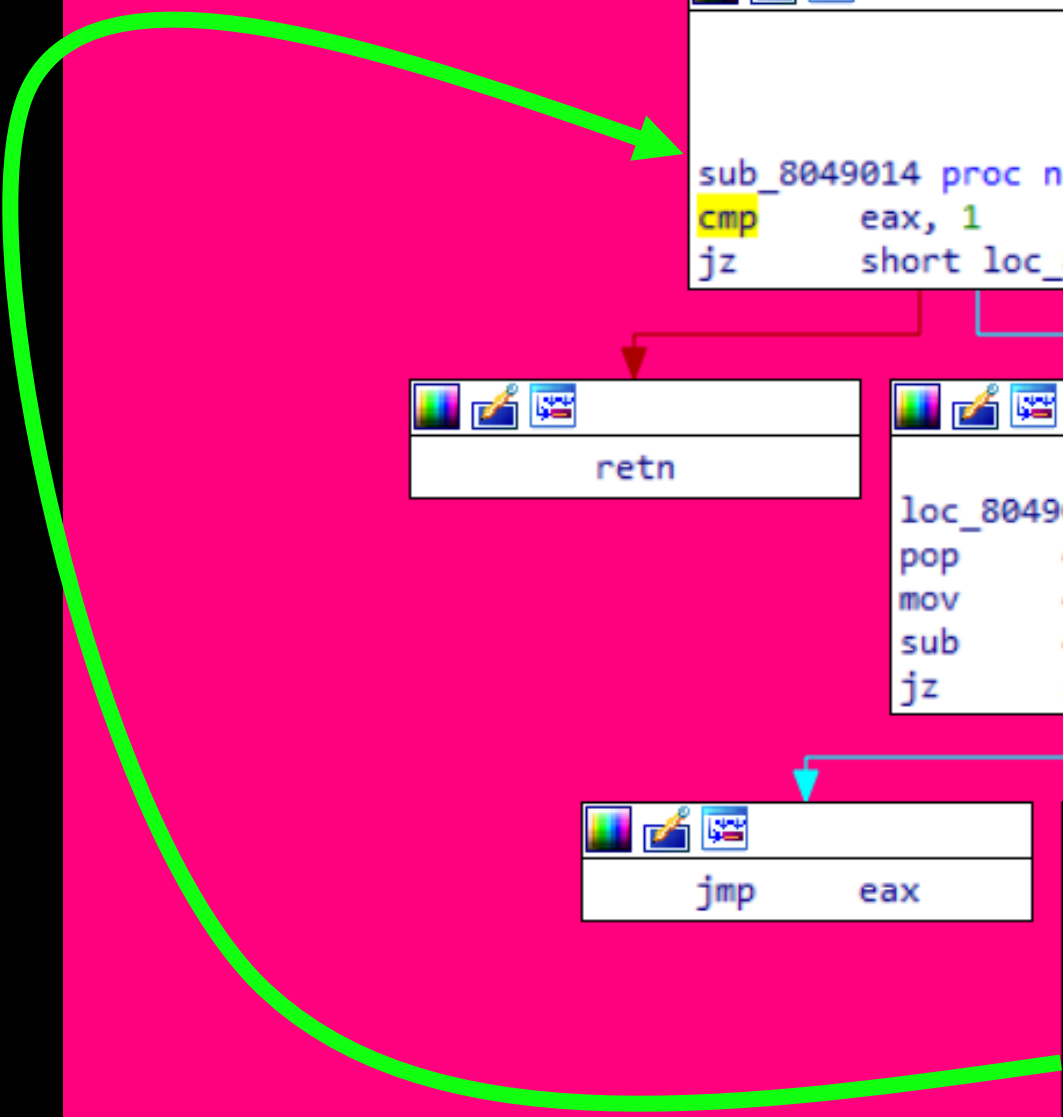
```
loc_804901A:  
pop    eax  
mov    eax, ds:dword_8087859[edi*4]  
sub    eax, 0ABAD1DEAh  
jz     short loc_804902B
```

```
jmp    eax
```

```
loc_804902B:  
mov    edi, 49Ah  
mov    eax, 1  
call   sub_8049014  
mov    eax, dword_8089014  
mov    dword_808900C, eax  
add    dword_808900C, 0FFFFFFh  
and    dword_808900C, 0FFFFFFh
```

Removes useless return address from stack  
Lookup address to jump to  
de-obfuscate address

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



AN  
EN

```
sub_8049014 proc near  
cmp    eax, 1  
jz     short loc_804901A
```

```
retn
```

```
loc_804901A:  
pop    eax  
mov    eax, ds:dword_8087859[edi*4]  
sub    eax, 0ABAD1DEAh  
jz     short loc_804902B
```

```
jmp    eax
```

```
loc_804902B:  
mov    edi, 49Ah  
mov    eax, 1  
call   sub_8049014  
mov    eax, dword_8089014  
mov    dword_808900C, eax  
add    dword_808900C, 0FFFFFFh  
and    dword_808900C, 0FFFFFFh
```

Removes useless return address from stack  
Lookup address to jump to  
de-obfuscate address

Jumps to correct location!!!

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



AN  
REN

```
sub_8049014 proc near  
  cmp     eax, 1  
  jz      short loc_804901A
```

```
retn
```

```
loc_804901A:  
  pop     eax  
  mov     eax, ds:dword_8087850[edi*4]  
  sub     eax, 0ABAD1DEAh  
  jz      short loc_804902B
```

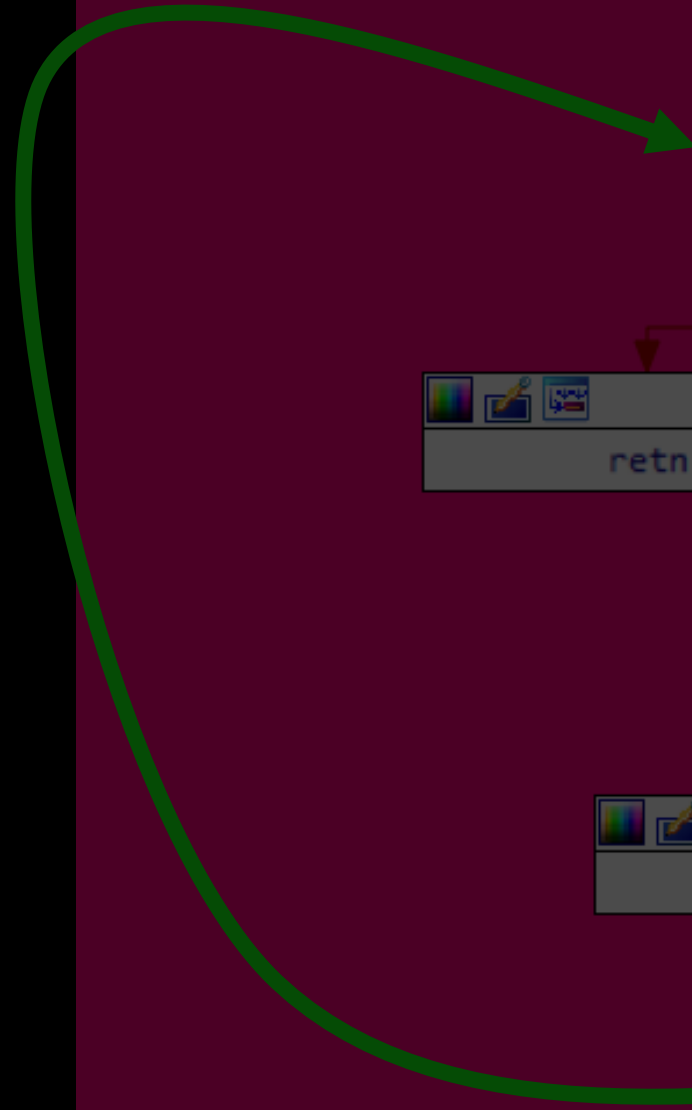
```
jmp     eax
```

```
loc_804902B:  
  mov     edi, 49Ah  
  mov     eax, 1  
  call    sub_8049014  
  mov     eax, dword_8089014  
  mov     dword_808900C, eax  
  add     dword_808900C, 0FFFFFFh  
  and     dword_808900C, 0FFFFFFh
```

Removes useless return address from stack  
Lookup address to jump to  
de-obfuscate address

Jumps to correct location!!!

Sets where to jump  
Sets "no don't exit program yet"  
Call itself



**Aside:**

```
sub_8049014 proc near
cmp     eax, 1
jz     short loc_804901A
```

**We need to obfuscate the address**

```
retn
```

```
loc_804901A:
pop     eax
mov     eax, ds:dword_8087850[edi*4]
sub     eax, 0ABAD1DEAh
jz     short loc_804902B
```

Removes useless return address from stack  
lookup address to jump to  
de-obfuscate address

**because otherwise IDA can find  
the jump target and draw an  
arrow to there, WHICH WE DONT  
WANT**

```
mov     dword_808900C, eax
add     dword_808900C, 0FFFFFFh
and     dword_808900C, 0FFFFFFh
```



**Result:**

```
sub_8049014 proc near  
cmp     eax, 1  
jz      short loc_804901A
```

**REST OF PROGRAM HAS NO CONTROLFLOW**

```
ds:dword_8087B59[edi*4]  
0ABAD1DEAh  
t loc_804902B
```

```
jmp     eax
```

```
loc_804902B:  
mov     edi, 49Ah  
mov     eax, 1  
call    sub_8049014  
mov     eax, dword_8089014  
mov     dword_808900C, eax  
add     dword_808900C, 0FFFFFFh  
and     dword_808900C, 0FFFFFFh  
mov     eax, dword_8089000
```



Jules 10/13/2022 12:37 PM

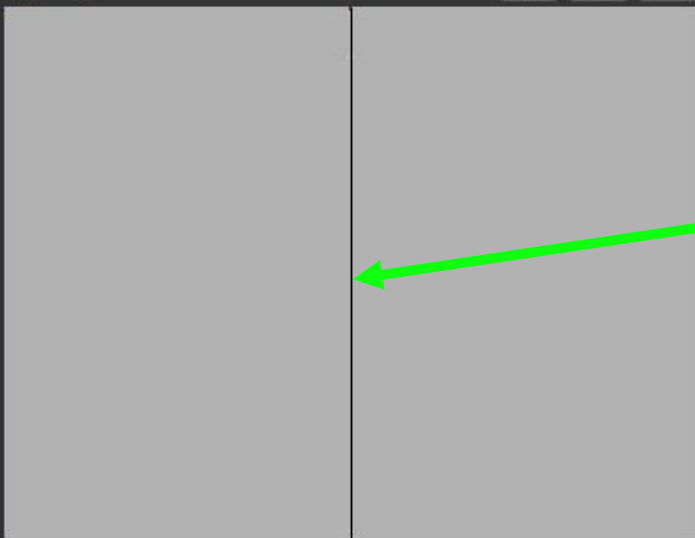
You've heard of controlflow flattening, get ready for controlflow lengthening



a.zip  
84.38 KB

```
/uwu$ ./a.art.exe  
> (defun uwu (x y) (+ x y))  
(lambda (x y) (+ x y))  
> (defun owo (x y z w) (uwu (uwu x y) (uwu z w)))  
(lambda (x y z w) (uwu (uwu x y) (uwu z w)))  
> (owo 1 2 3 4)  
10  
> |
```

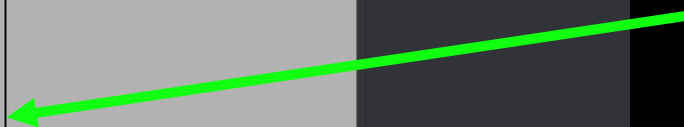
Graph overview



Lisp



This is  
the  
graph!



enigmatrix 10/13/2022 12:40 PM

HAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHAHA

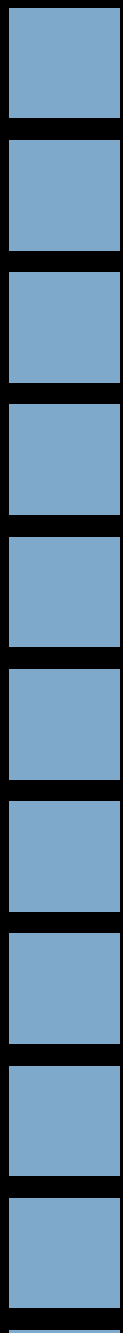
u did it

madlad

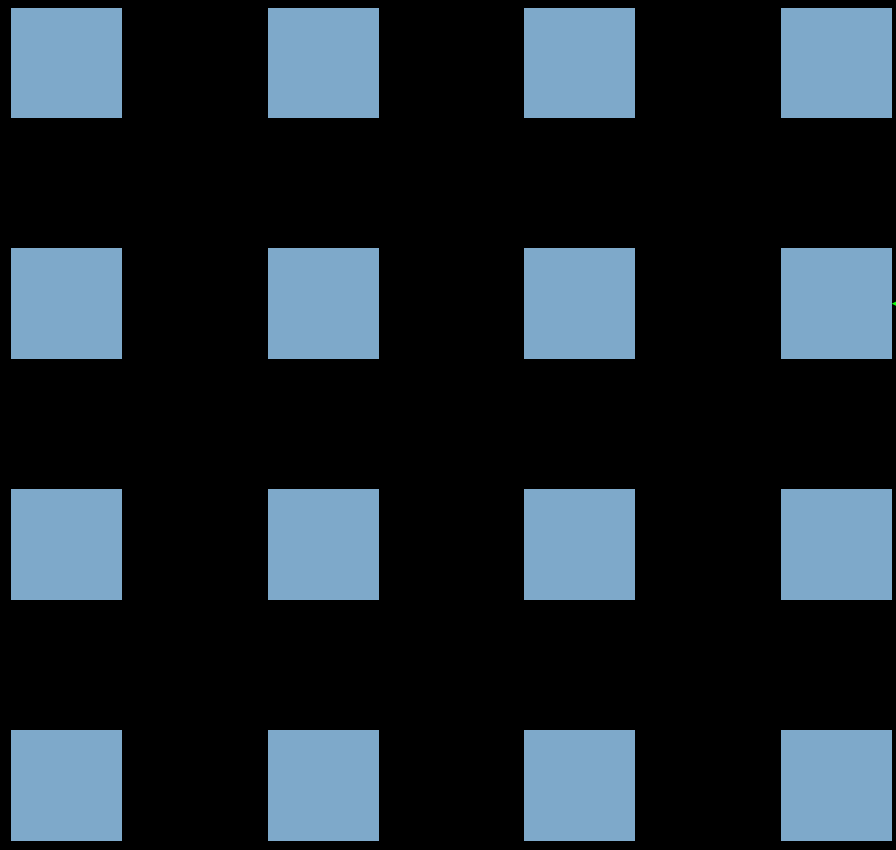


**All the code**

**All the code  
Cut them up**



All the code  
Cut them up  
Rearrange into a picture



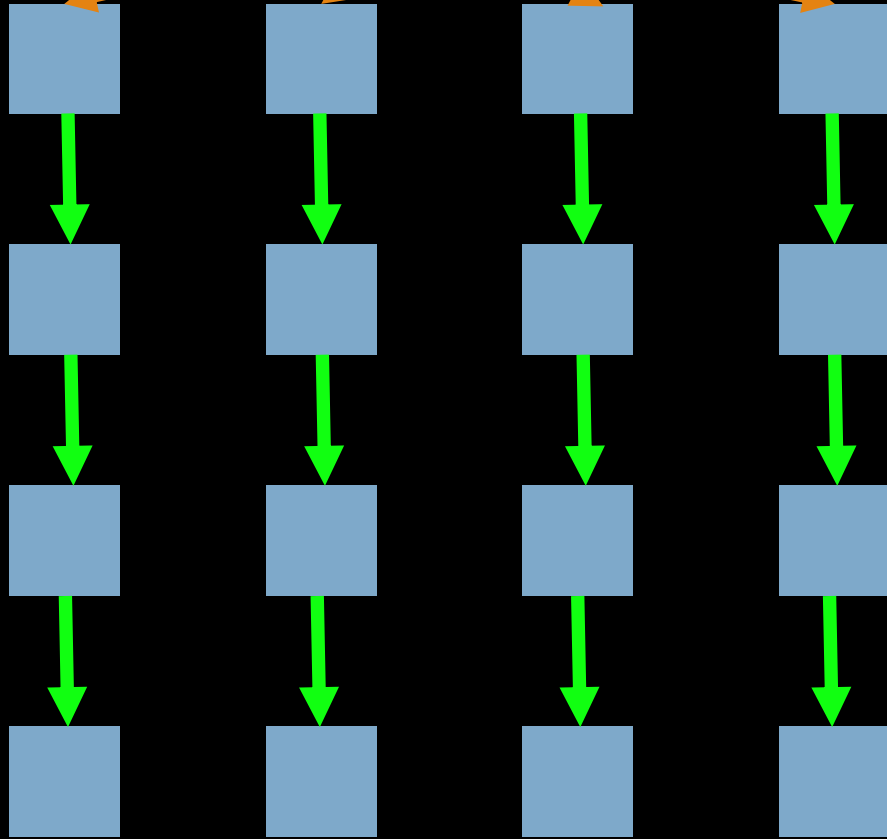
Brightness of  
pixel given by  
length of each  
"piece"



Handles all  
actual  
controlflow

Preamble

Fake switch case



Fake controlflow

All the code  
Cut them up  
Rearrange into a picture  
Connect by adding **jumps**

# Lisp

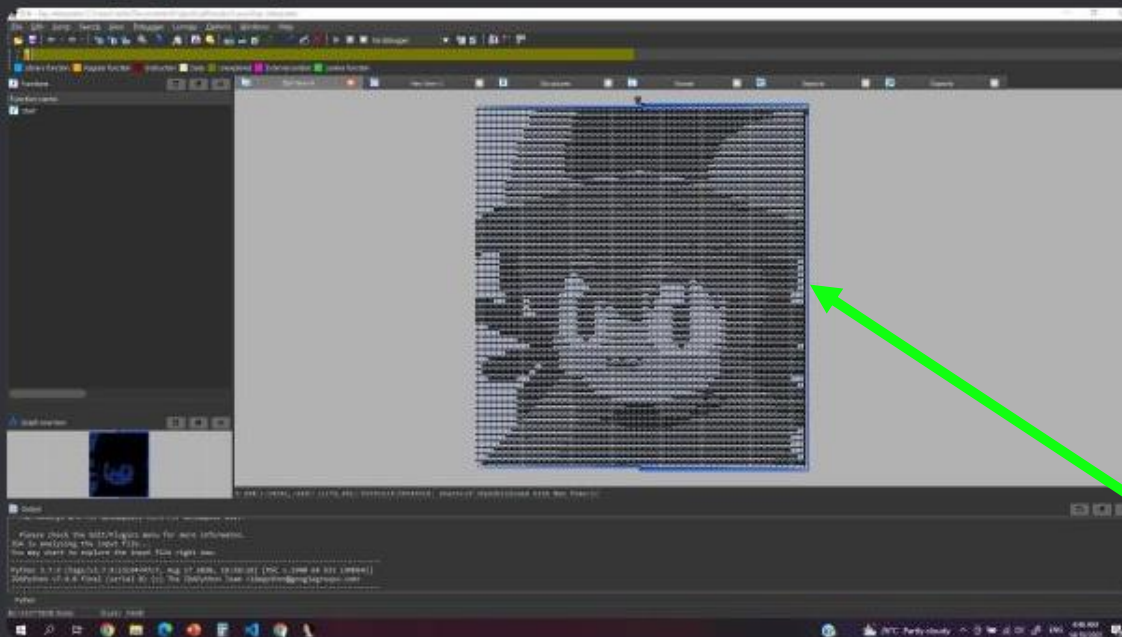


Jules 10/14/2022 4:43 AM



[lisp-interpreter.zip](#)  
113.01 KB

Ok am going to sleep



6



enigmatrix 10/14/2022 9:09 AM

..



sansders 10/14/2022 10:24 AM

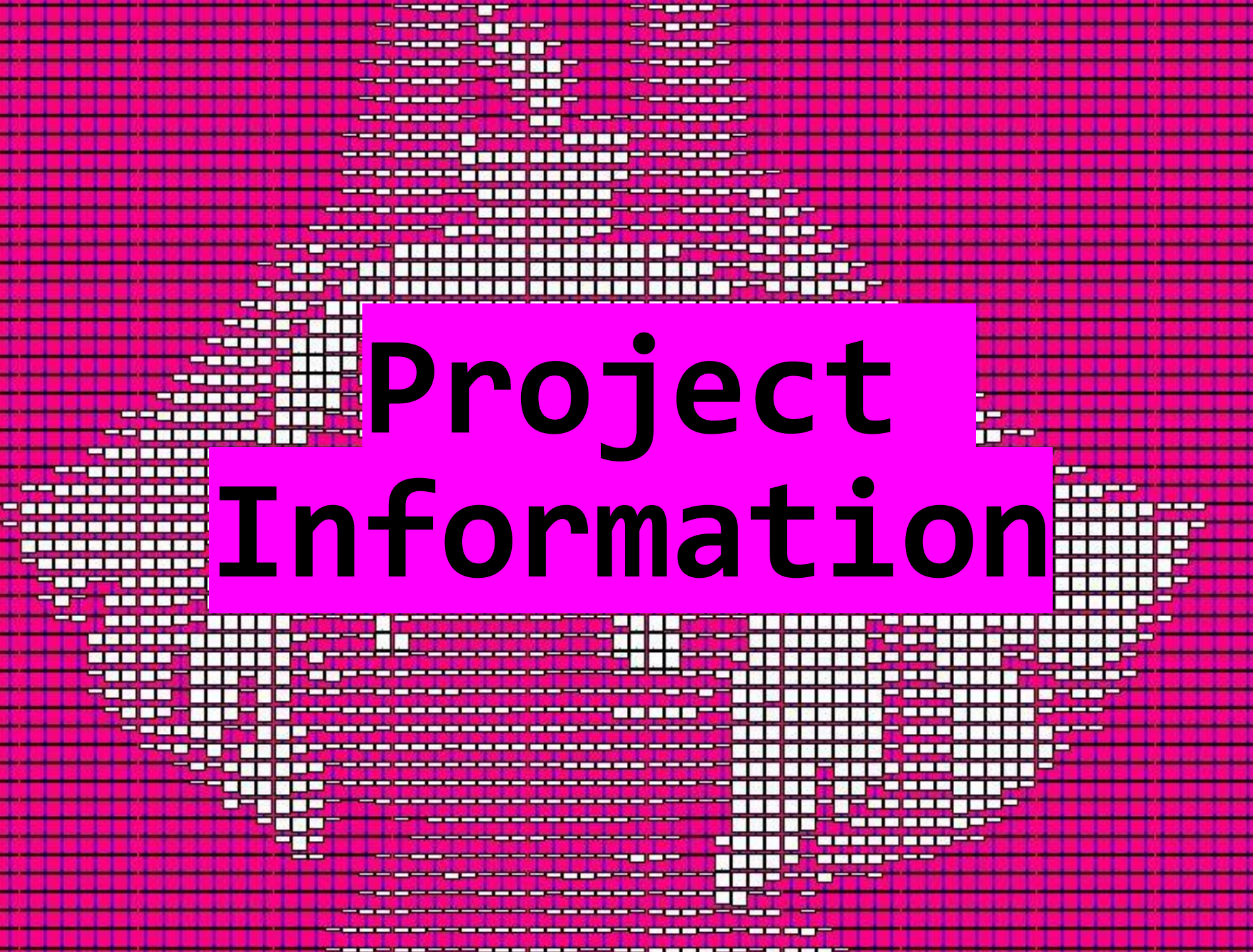
oh wow you can potentially do bad apple on ida  
oh nvm you already did LOL



ariana 10/14/2022 10:50 AM

and this is how re challenges are made at 5am

# This is the graph!



# **Project Information**



# Documentation for Artfuscator



[github.com/JuliaPoo/artfuscator](https://github.com/JuliaPoo/artfuscator)

# Documentation for Artfuscator

github.com/JuliaPoo/artfuscator

```
(lambda (x y z w) (u (u x y) (u z w)))  
> (v 1 2 3 4)  
10  
> █
```

[How it works](#)

TODO

[Credits](#)

- [Christopher Domas](#) for the original idea and work on REPsych
- [Shinh](#) for ELVM, which is an awesome project

**EVERYTHING YOU NEED IS HERE**

# Licensing

---

The MIT License does not hold me  
LIABLE FOR ANY CLAIM, DAMAGES, PSYCHOLOGICAL DAMAGES OR OTHER  
LIABILITY,  
ARISING FROM THE USE OF THIS PROJECT

## Limitations

- × Liability
- × Warranty

A pixelated globe of the Earth is centered on a black background. The globe is composed of a grid of small squares, with white squares forming the continents and black squares forming the oceans. The globe is slightly tilted. In the center of the globe, there is a red rectangular box containing the word "DEMO" in white, bold, sans-serif capital letters.

**DEMO**

re: important information  
about me

---

4. HMU

e0958745@u

.nus.edu