REpsych

: psycholigical warfare in reverse engineering

{ def con 2015 // domas

- This serves no purpose

Warning

- Taking something apart …

  … to figure out how it works
- With software…
  - Interfacing
  - Documentation
  - Obsolescence
  - Bug fixing
  - Academic

Reverse Engineering?

- Taking something apart …

  … to figure out how it works
- With software…
  - Military/commercial espionage
  - Unauthorized duplication
  - Security analysis
  - Vulnerability analysis
  - Malware analysis

Reverse Engineering?

- Whenever we write something awesome…
  - Video game
  - Encryption algorithm
  - Malware
  - 0-Day
  - RAT
- … someone, at some point, is going to …
  - Capture it
  - Dissect it
  - Reverse it

Reverse Engineering?

- If you don't want your work destroyed …
  … it pays to plan ahead

Anti-RE

Anti-RE

- Encryption
- Obfuscation
- Anti-debugging

& objdump –d –Mintel a.out

Reverse Engineering.

```
4004e9:  mov    DWORD PTR [rbp-0x8],0x0
4004f2:  push   600004
4004f8:  call   printf
4004fa:  pop    eax
4004fc:  add    DWORD PTR [rbp-0x8],0x1
400500:  cmp    DWORD PTR [rbp-0x8],0x100
400507:  jle    4004f2 <main+0xb>
```

mov

- mov is Turing-complete
  - Stephen Dolan
  - http://www.cl.cam.ac.uk/~sd601/papers/mov.pdf

- mov destination, source

mov

- Any code we write …
- … can be written as a set of movs instead
- … *and nothing else*
- *Really?*
- That'd be tough to reverse engineer, wouldn't it?

Turing Complete?

```
4004e9:  mov    DWORD PTR [rbp-0x8],0x0
4004f2:  push   600004
4004f8:  call   printf
4004fa:  pop    eax
4004fc:  add    DWORD PTR [rbp-0x8],0x1
400500:  cmp    DWORD PTR [rbp-0x8],0x100
400507:  jle    4004f2 <main+0xb>
```

```
80515bc:  mov    eax,ds:0x835d81a
80515c1:  mov    ebx,DWORD PTR [eax+0x835d6fc]
80515c7:  mov    edx,DWORD PTR ds:0x835d7da
80515cd:  mov    eax,0x0
80515d2:  mov    al,BYTE PTR [ebx+edx*1]
80515d5:  mov    al,BYTE PTR [eax+0x835dc7e]
80515db:  mov    BYTE PTR [ebx+edx*1],al
80515de:  mov    eax,ds:0x835d81a
80515e3:  mov    ebx,DWORD PTR [eax+0x835d6fc]
80515e9:  mov    edx,DWORD PTR ds:0x835d7da
80515ef:  mov    eax,0x0
80515f4:  mov    al,BYTE PTR [ebx+edx*1]
```

- mov-only C Compiler
  - https://github.com/xoreaxeaxeax
- First single instruction C compiler!

The M/o/Vfuscator

- factor 20460
- prime
- decss
- Lost
- M/o/Vfuscator

The M/o/Vfuscator

- Crackmes

The M/o/Vfuscator

- How would an experienced reverse engineer approach this?

```
mov [dword 0x80a0451],edx
mov eax,0x0
mov ax,[0x80a0451]
mov byte [eax+0x80e17bc],0x0
mov al,[eax+0x80e17bc]
mov [0x80a0451],al
mov eax,[0x80a0556]
mov edx,[eax+0x80a058e]
mov eax,[0x80a0451]
mov eax,[eax+edx]
mov [0x80a044d],eax
mov eax,[0x80a044d]
mov eax,[eax+0x80a054e]
mov dword [eax],0x139
mov eax,[0x80a044d]
mov eax,[eax+0x80a055e]
mov dword [eax],0x0
mov eax,[0x80a044d]
mov eax,[eax+0x80a056e]
mov dword [eax],0x4
mov eax,[0x80a0556]
mov eax,[eax+0x80a05a6]
mov [0x80a0451],eax
mov eax,0x0
mov ax,[0x80a0546]
mov byte [eax+0x80e17bc],0x0
mov al,[eax+0x80e17bc]
mov [0x80a044d],al
mov eax,[0x80a044d]
mov edx,[eax+0x80a058e]
mov eax,[0x80a0451]
mov eax,[eax+edx]
mov [0x80a044d],eax
mov eax,[0x80a0566]
mov eax,[eax+0x80a05a6]
mov [0x80a0451],eax
mov eax,[0x80a044d]
mov edx,[eax+0x80a058e]
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a051e]
mov eax,[ebx]
mov edx,0x0
mov dx,[eax+eax+0x80c0bba]
mov [ebx],edx
mov eax,[0x80a0556]
mov ebx,[eax+0x80a0438]
mov edx,[dword 0x80a0516]
mov eax,0x0
mov al,[ebx+edx]
mov al,[eax+0x80a09ba]
mov edx,[eax+0x80a058e]
mov eax,[0x80a0451]
```


‘nope’

- Anti-RE
  - Code doesn't have to be hard to reverse
  - Just need to make the reverser give up

Realization

- Demoralization
  - Break down the reverser

Psychological Warfare

- How else can we make a reverser quit?

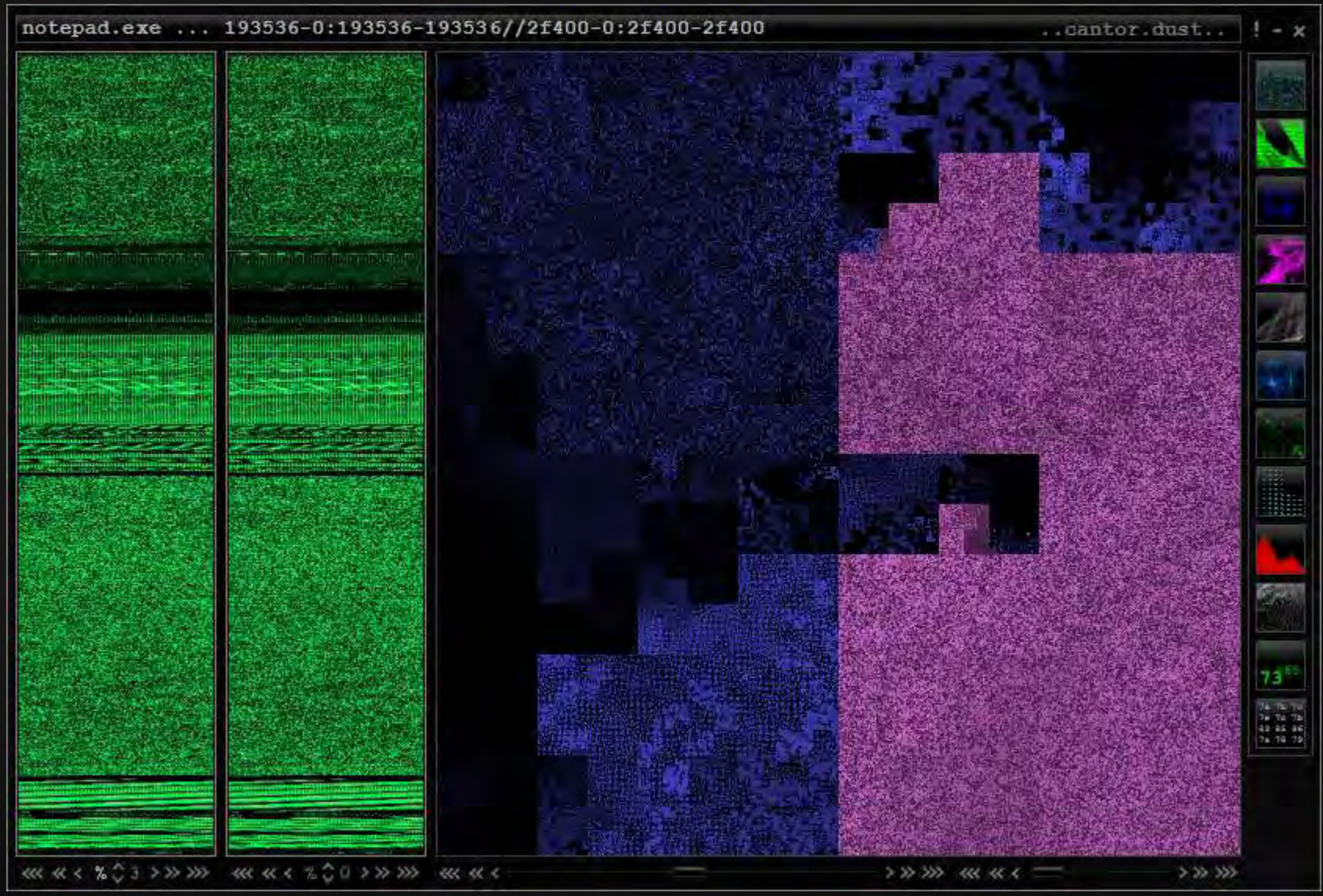
Psychological Warfare

Sending messages…

..cantor.dust..

- Visualize data patterns
  - Default: entropy distribution

..cantor.dust..


notepad.exe ... 193536-0:193536-193536//2f400-0:2f400-2f400
..cantor.dust..
! - x

- Send a message?
  - Run a message through an inverse Hilbert transform
  - Rebuild program to match desired entropy

..cantor.dust..

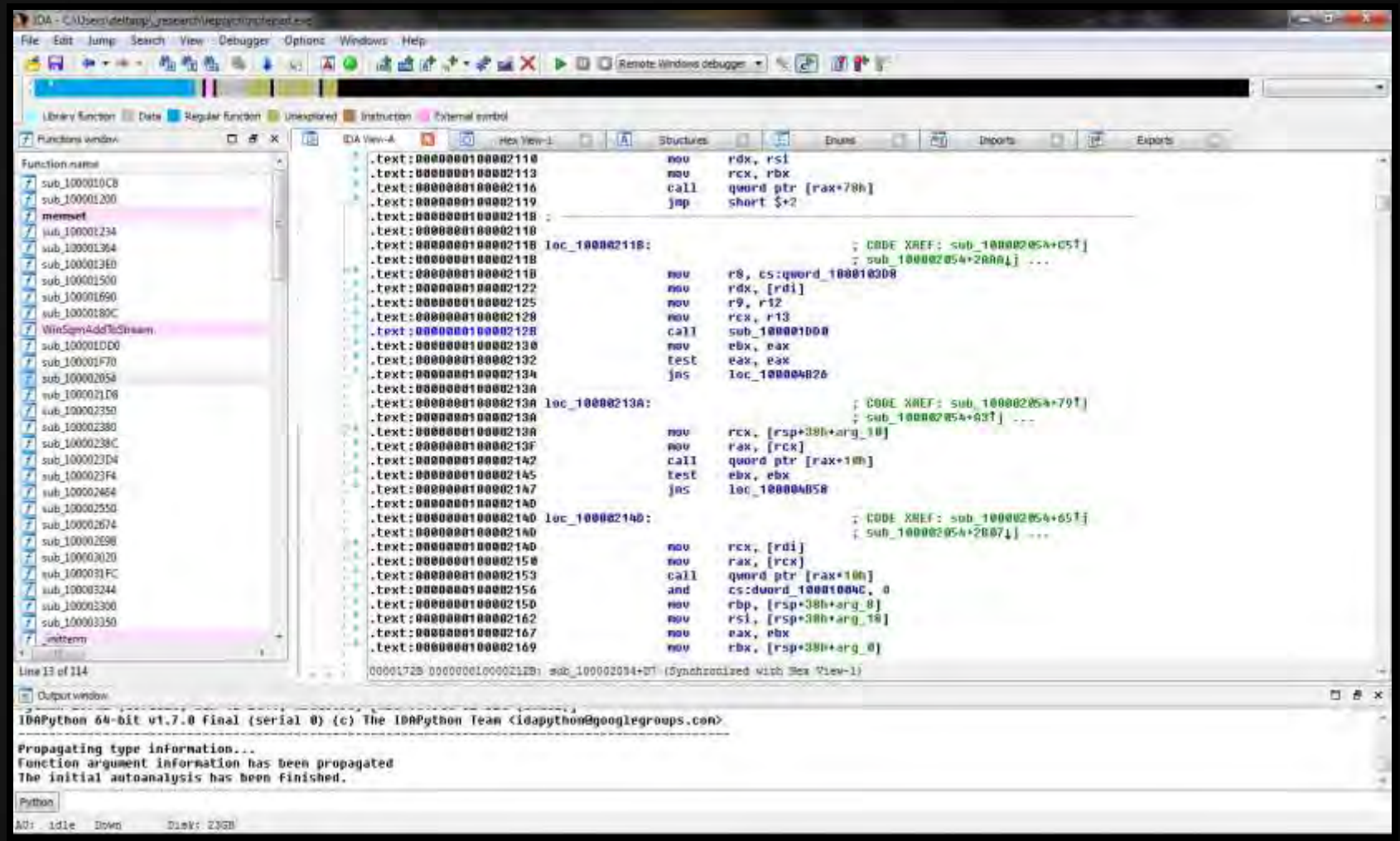

HelloWorld.exe ... 16777216-0:16777216-16777216//1000000-0:1000000-1000000
..cantor.dust..
Top o' the
mornin'
to ya!

- Strings?

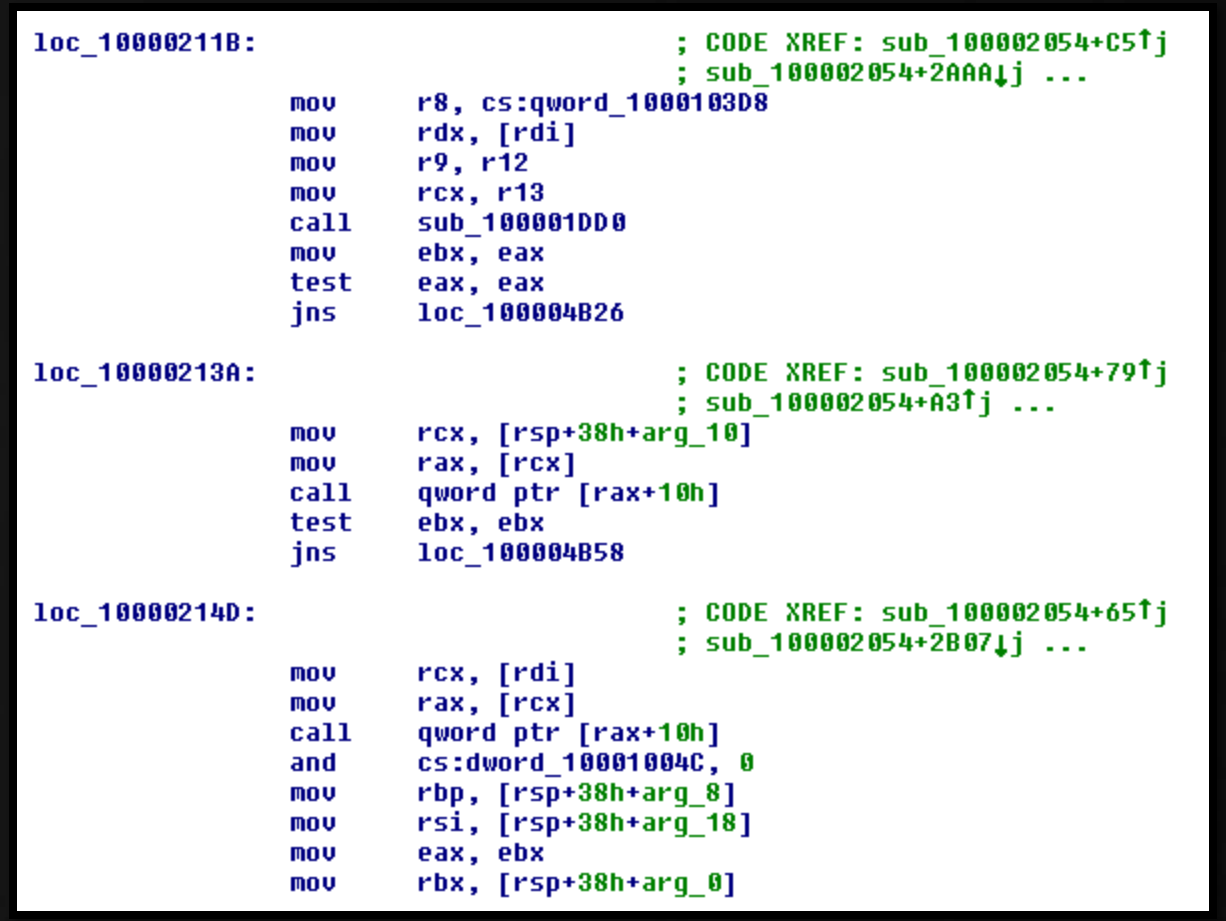
Sending messages

- These are horrible…
  - No one will ever see the message
  - And if they do, they won't care
  - Need something better…

Sending messages


File Edit Jump Search View Debugger Options Windows Help
Remote Windows debugger
Library function Data Regular function Unexplored Instruction External symbol
Functions window
Function name
sub_1000010CB
sub_100001200
memset
sub_100001234
sub_100001364
sub_1000013E0
sub_100001500
sub_100001690
sub_1000018DC
WinSqmAddToStream
sub_100001DD0
sub_100001F70
sub_100002054
sub_1000021D8
sub_100002350
sub_100002380
sub_1000023BC
sub_1000023D4
sub_1000023F4
sub_100002464
sub_100002550
sub_100002674
sub_100002698
sub_100003020
sub_1000031FC
sub_100003244
sub_100003300
sub_100003350
_initterm
Line 13 of 114
IDA View-A
Hex View-1
Structures
Enums
Imports
Exports
.text:0000000100002110 mov rdx, rsi
.text:0000000100002113 mov rcx, rbx
.text:0000000100002116 call qword ptr [rax+78h]
.text:0000000100002119 jmp short $+2
.text:000000010000211B ;
.text:000000010000211B
.text:000000010000211B loc_10000211B: ; CODE XREF: sub_100002054+C5↑j
.text:000000010000211B ; sub_100002054+2AA↓j ...
.text:000000010000211B mov r8, cs:qword_100010308
.text:0000000100002122 mov rdx, [rdi]
.text:0000000100002125 mov r9, r12
.text:0000000100002128 mov rcx, r13
.text:000000010000212B call sub_100001DD0
.text:0000000100002130 mov ebx, eax
.text:0000000100002132 test eax, eax
.text:0000000100002134 jns loc_100004B26
.text:000000010000213A
.text:000000010000213A loc_10000213A: ; CODE XREF: sub_100002054+79↑j
.text:000000010000213A ; sub_100002054+83↑j ...
.text:000000010000213A mov rcx, [rsp+38h+arg_10]
.text:000000010000213F mov rax, [rcx]
.text:0000000100002142 call qword ptr [rax+10h]
.text:0000000100002145 test ebx, ebx
.text:0000000100002147 jns loc_100004B58
.text:000000010000214D
.text:000000010000214D loc_10000214D: ; CODE XREF: sub_100002054+65↑j
.text:000000010000214D ; sub_100002054+2B7↓j ...
.text:000000010000214D mov rcx, [rdi]
.text:0000000100002150 mov rax, [rcx]
.text:0000000100002153 call qword ptr [rax+10h]
.text:0000000100002156 and cs:dword_10001004C, 0
.text:000000010000215D mov rbp, [rsp+38h+arg_8]
.text:0000000100002162 mov rsi, [rsp+38h+arg_10]
.text:0000000100002167 mov eax, ebx
.text:0000000100002169 mov rbx, [rsp+38h+arg_0]
00001728 0000000100002128: sub_100002054+D4 (Synchronized with Hex View-1)
Output window
IDAPython 64-bit v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished.
Python
AU: idle Down Disk: 23GB

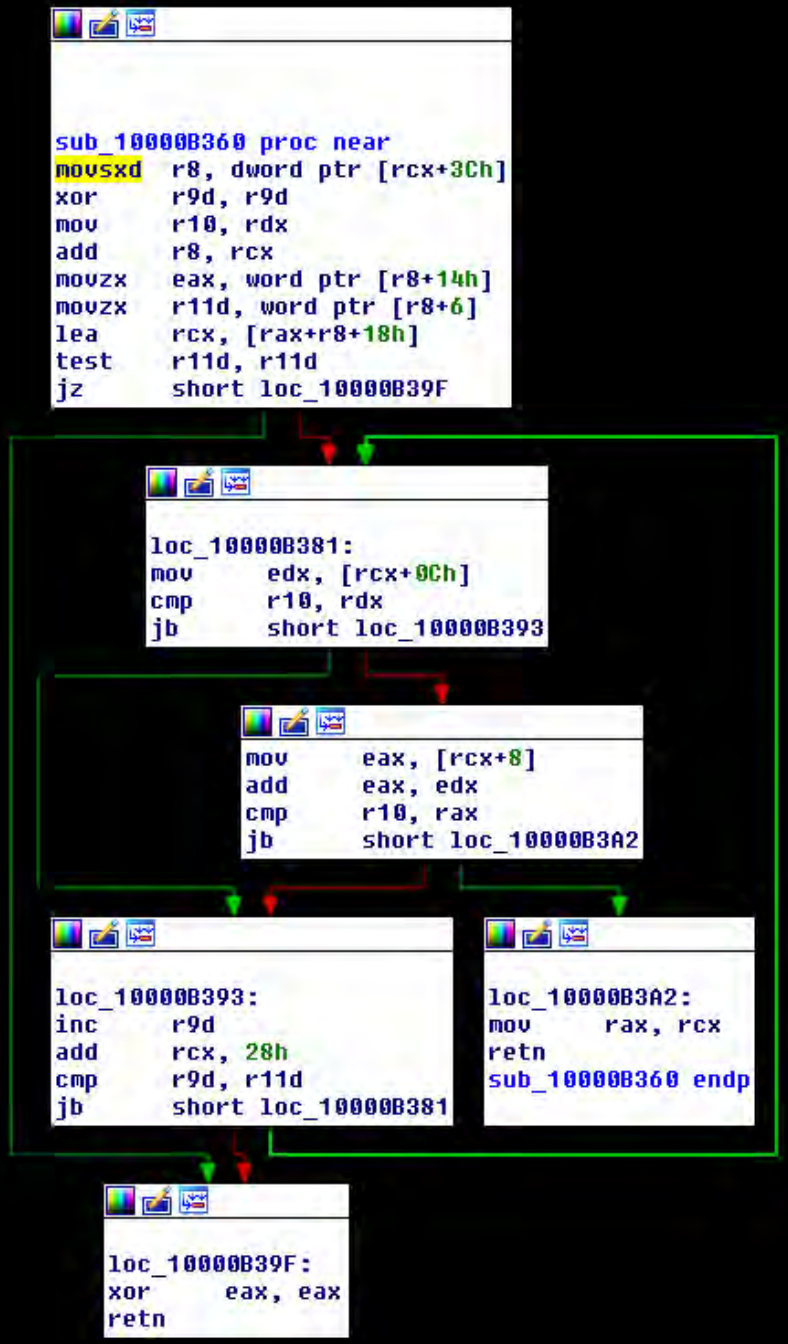
IDA


loc_10000211B: ; CODE XREF: sub_100002054+C5↑j
; sub_100002054+2AAA↓j ...
mov r8, cs:qword_1000103D8
mov rdx, [rdi]
mov r9, r12
mov rcx, r13
call sub_100001DD0
mov ebx, eax
test eax, eax
jns loc_100004B26
loc_10000213A: ; CODE XREF: sub_100002054+79↑j
; sub_100002054+A3↑j ...
mov rcx, [rsp+38h+arg_10]
mov rax, [rcx]
call qword ptr [rax+10h]
test ebx, ebx
jns loc_100004B58
loc_10000214D: ; CODE XREF: sub_100002054+65↑j
; sub_100002054+2B07↓j ...
mov rcx, [rdi]
mov rax, [rcx]
call qword ptr [rax+10h]
and cs:dword_10001004C, 0
mov rbp, [rsp+38h+arg_8]
mov rsi, [rsp+38h+arg_18]
mov eax, ebx
mov rbx, [rsp+38h+arg_0]

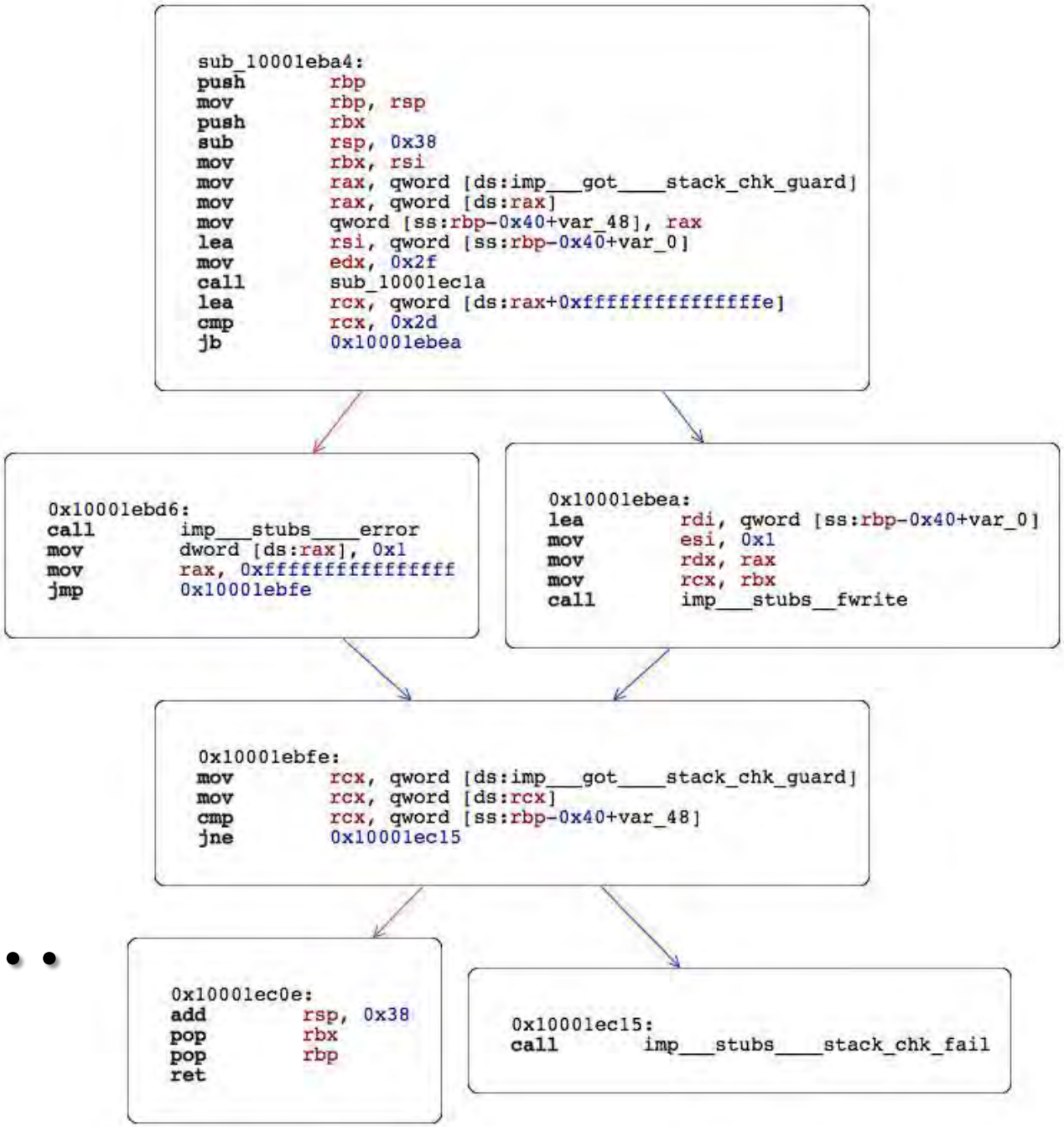
IDA

Control flow graphs…

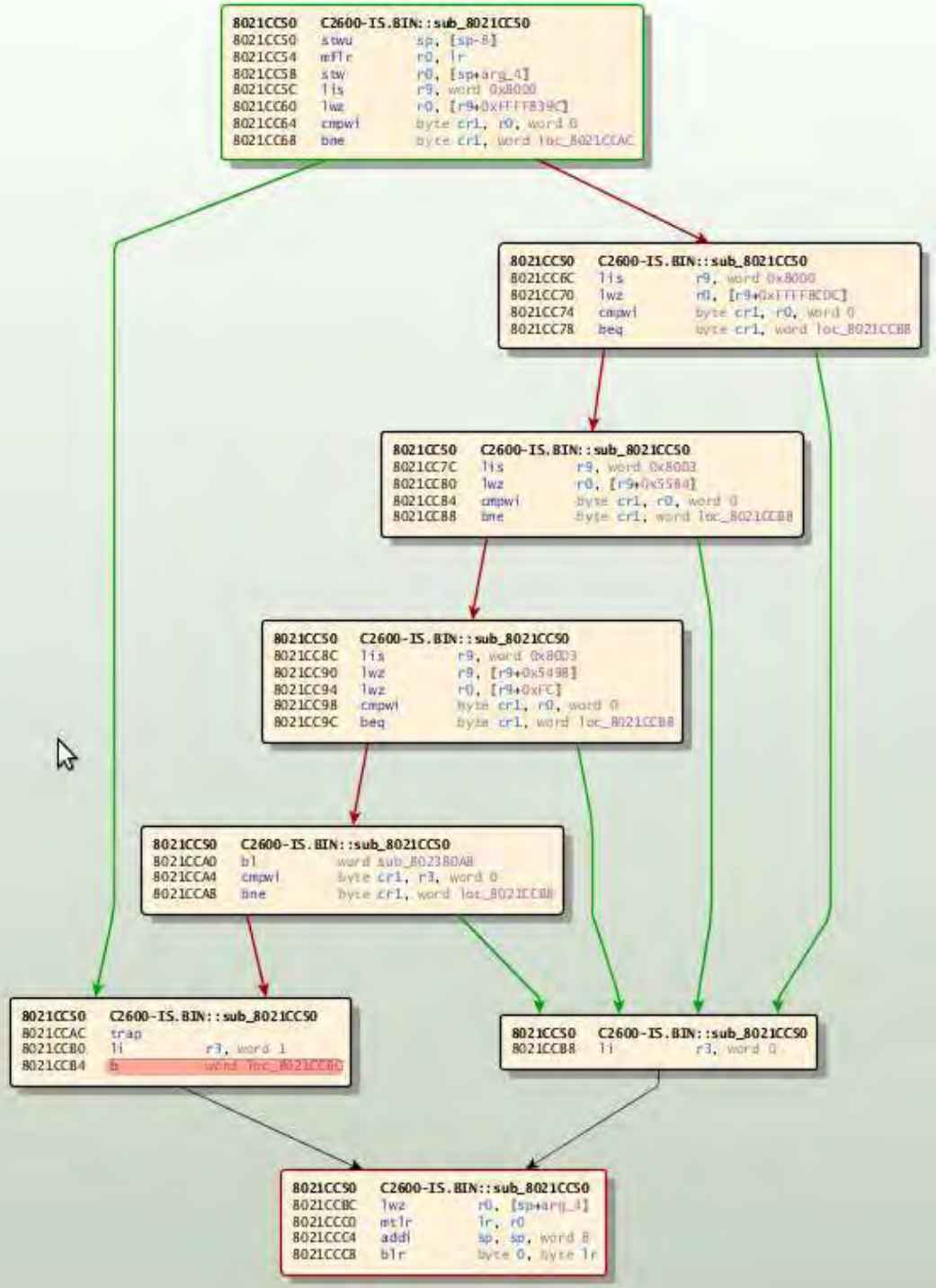
IDA…


sub_10000B360 proc near
movsxd r8, dword ptr [rcx+3Ch]
xor r9d, r9d
mov r10, rdx
add r8, rcx
movzx eax, word ptr [r8+14h]
movzx r11d, word ptr [r8+6]
lea rcx, [rax+r8+18h]
test r11d, r11d
jz short loc_10000B39F
loc_10000B381:
mov edx, [rcx+0Ch]
cmp r10, rdx
jb short loc_10000B393
mov eax, [rcx+8]
add eax, edx
cmp r10, rax
jb short loc_10000B3A2
loc_10000B393:
inc r9d
add rcx, 28h
cmp r9d, r11d
jb short loc_10000B381
loc_10000B3A2:
mov rax, rcx
retn
sub_10000B360 endp
loc_10000B39F:
xor eax, eax
retn

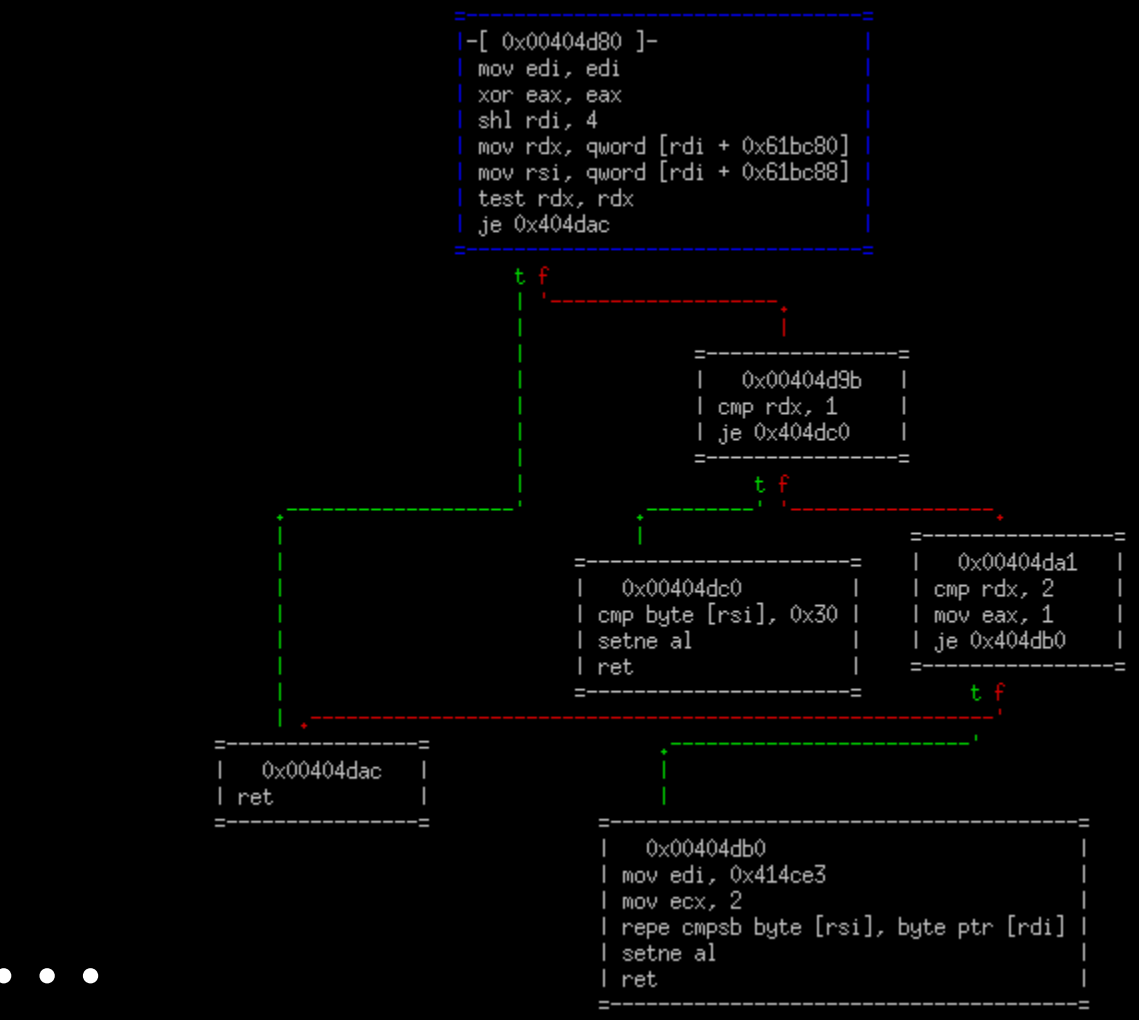
Hopper…


sub_10001eba4:
push rbp
mov rbp, rsp
push rbx
sub rsp, 0x38
mov rbx, rsi
mov rax, qword [ds:imp___got____stack_chk_guard]
mov rax, qword [ds:rax]
mov qword [ss:rbp-0x40+var_48], rax
lea rsi, qword [ss:rbp-0x40+var_0]
mov edx, 0x2f
call sub_10001ec1a
lea rcx, qword [ds:rax+0xfffffffffffffffe]
cmp rcx, 0x2d
jb 0x10001ebea
0x10001ebd6:
call imp___stubs____error
mov dword [ds:rax], 0x1
mov rax, 0xffffffffffffffff
jmp 0x10001ebfe
0x10001ebea:
lea rdi, qword [ss:rbp-0x40+var_0]
mov esi, 0x1
mov rdx, rax
mov rcx, rbx
call imp___stubs__fwrite
0x10001ebfe:
mov rcx, qword [ds:imp___got____stack_chk_guard]
mov rcx, qword [ds:rcx]
cmp rcx, qword [ss:rbp-0x40+var_48]
jne 0x10001ec15
0x10001ec0e:
add rsp, 0x38
pop rbx
pop rbp
ret
0x10001ec15:
call imp___stubs____stack_chk_fail

BinNavi…


8021CC50 C2600-IS.BIN::sub_8021CC50
8021CC50 stwu sp, [sp-8]
8021CC54 mflr r0, lr
8021CC58 stw r0, [sp+arg_4]
8021CC5C lis r9, word 0x8000
8021CC60 lwz r0, [r9+0xFFFF834C]
8021CC64 cmpwi byte cr1, r0, word 0
8021CC68 bne byte cr1, word loc_8021CCAC
8021CC50 C2600-IS.BIN::sub_8021CC50
8021CC6C lis r9, word 0x8000
8021CC70 lwz r0, [r9+0xFFFF8CDC]
8021CC74 cmpwi byte cr1, r0, word 0
8021CC78 beq byte cr1, word loc_8021CCB8
8021CC50 C2600-IS.BIN::sub_8021CC50
8021CC7C lis r9, word 0x8003
8021CC80 lwz r0, [r9+0x5584]
8021CC84 cmpwi byte cr1, r0, word 0
8021CC88 bne byte cr1, word loc_8021CCB8
8021CC50 C2600-IS.BIN::sub_8021CC50
8021CC8C lis r9, word 0x8003
8021CC90 lwz r9, [r9+0x5498]
8021CC94 lwz r0, [r9+0xFC]
8021CC98 cmpwi byte cr1, r0, word 0
8021CC9C beq byte cr1, word loc_8021CCB8
8021CCA0 bl word sub_80238DA8
8021CCA4 cmpwi byte cr1, r3, word 0
8021CCA8 bne byte cr1, word loc_8021CCB8
8021CCAC trap
8021CCB0 li r3, word 1
8021CCB4 b word loc_8021CCBC
8021CCB8 li r3, word 0
8021CCBC lwz r0, [sp+arg_4]
8021CCC0 mtlr lr, r0
8021CCC4 addi sp, sp, word 8
8021CCC8 blr byte 0, byte lr

Radare…

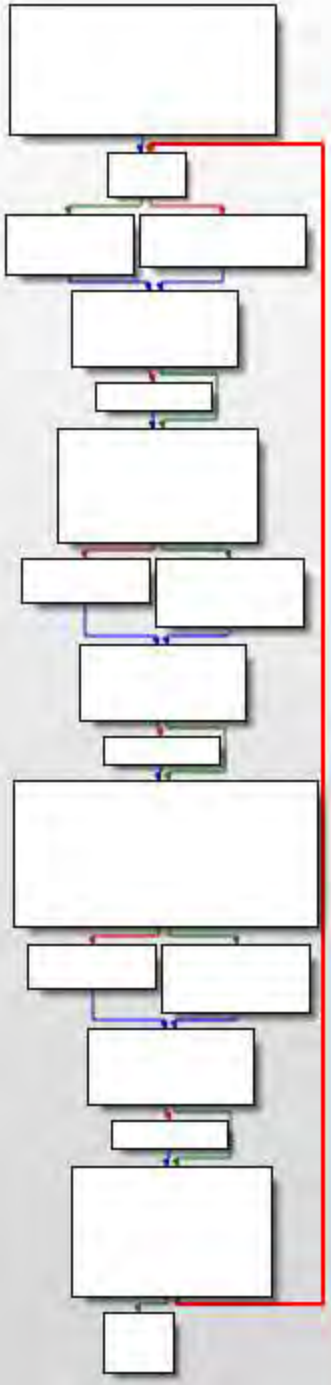

-[0x00404d80]-
mov edi, edi
xor eax, eax
shl rdi, 4
mov rdx, qword [rdi + 0x61bc80]
mov rsi, qword [rdi + 0x61bc88]
test rdx, rdx
je 0x404dac
t f
0x00404d9b
cmp rdx, 1
je 0x404dc0
t f
0x00404dc0
cmp byte [rsi], 0x30
setne al
ret
0x00404da1
cmp rdx, 2
mov eax, 1
je 0x404db0
t f
0x00404dac
ret
0x00404db0
mov edi, 0x414ce3
mov ecx, 2
repe cmpsb byte [rsi], byte ptr [rdi]
setne al
ret

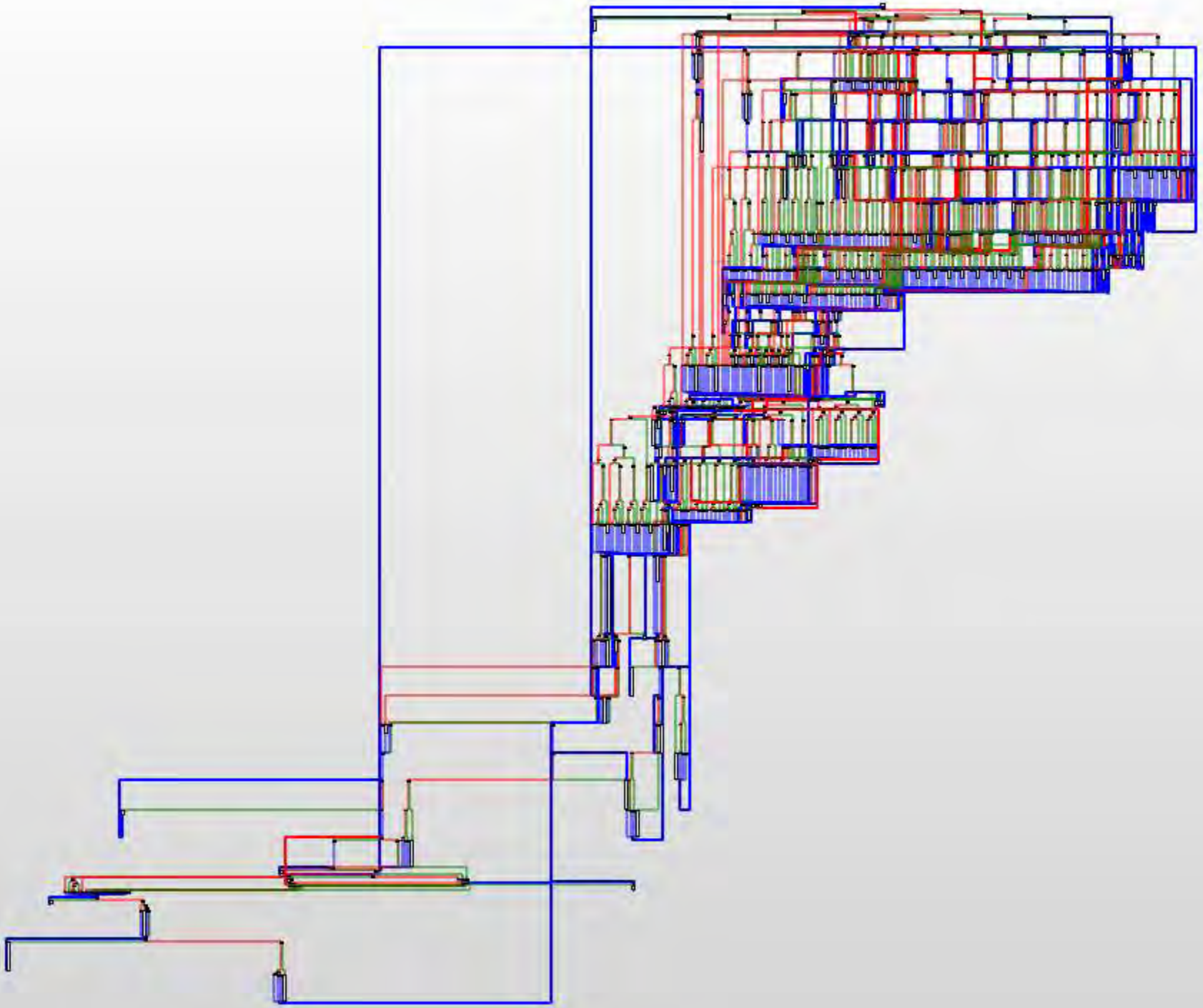
- We'll look at IDA
- But the algorithm will work on anything

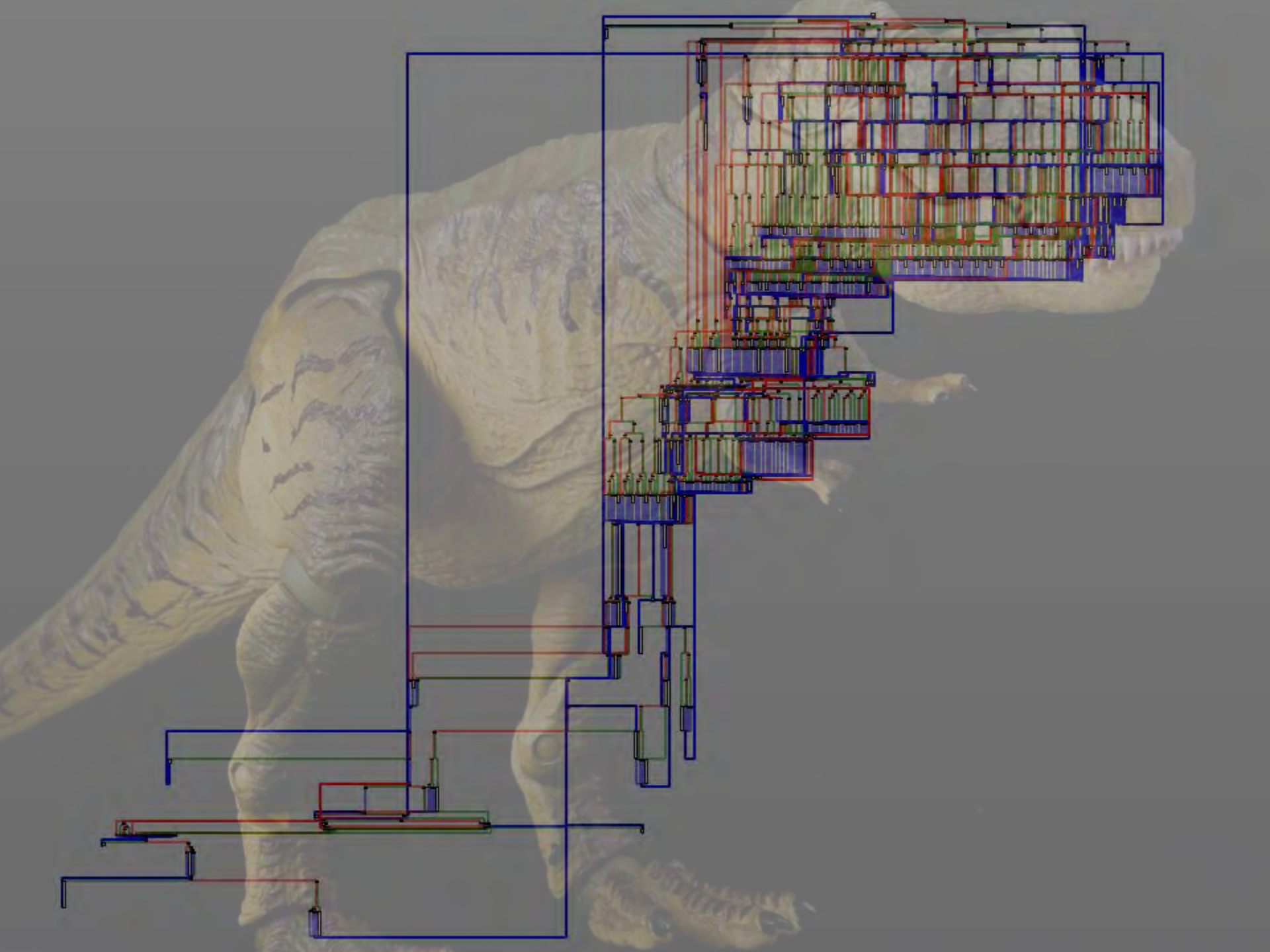
IDA

- If you stare at these control graphs long enough…
  … they almost start to look like things

Idea…





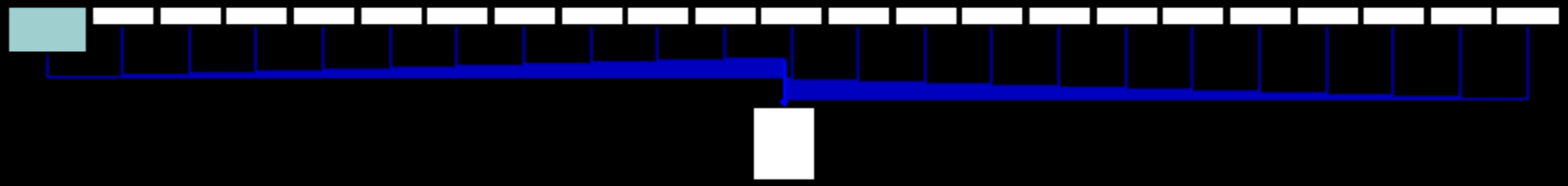


- Could we send a message through a CFG?
- Reverse engineer IDA?
  - Yep!

Drawing with CFGs

- Draw horizontal lines:
  - Switch
  - “Orphan” jumps
    - `jmp a`
    - `jmp a`
    - `jmp a`
    - `jmp a`
    - `jmp a`
    - `jmp a`
    - `a:`

Idea 1



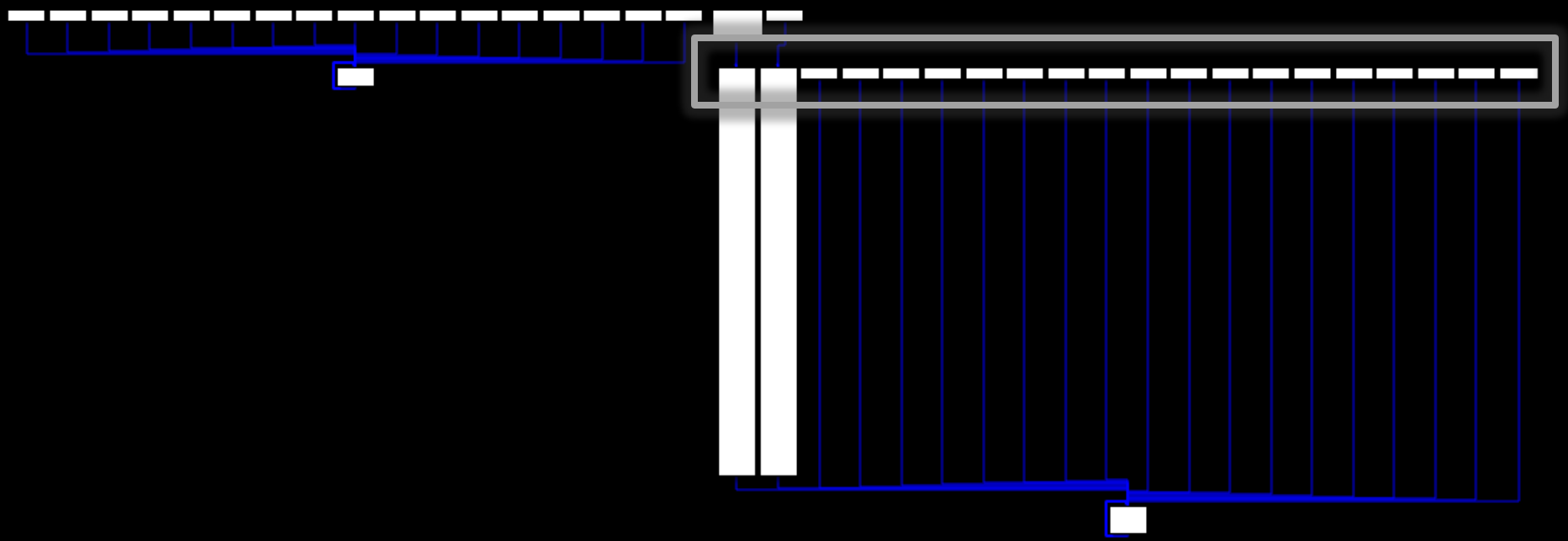
- Draw vertical lines:
  - Non-branching code
    - nop
    - nop
    - nop
    - nop
    - nop
    - nop

Idea 1

- Combining the two
  - Etch-a-sketch, in IDA!

Idea 1

```
top:
jmp left
jmp top_end
… ; repeat
jmp right_side
top_end:
jmp $

left_side:
nop
… ; repeat
jmp bottom_left

right_side:
nop
… ; repeat
jmp bottom_right

bottom:
botton_left:
jmp bottom_end
… ; repeat
bottom_right:
bottom_end:
ret
```

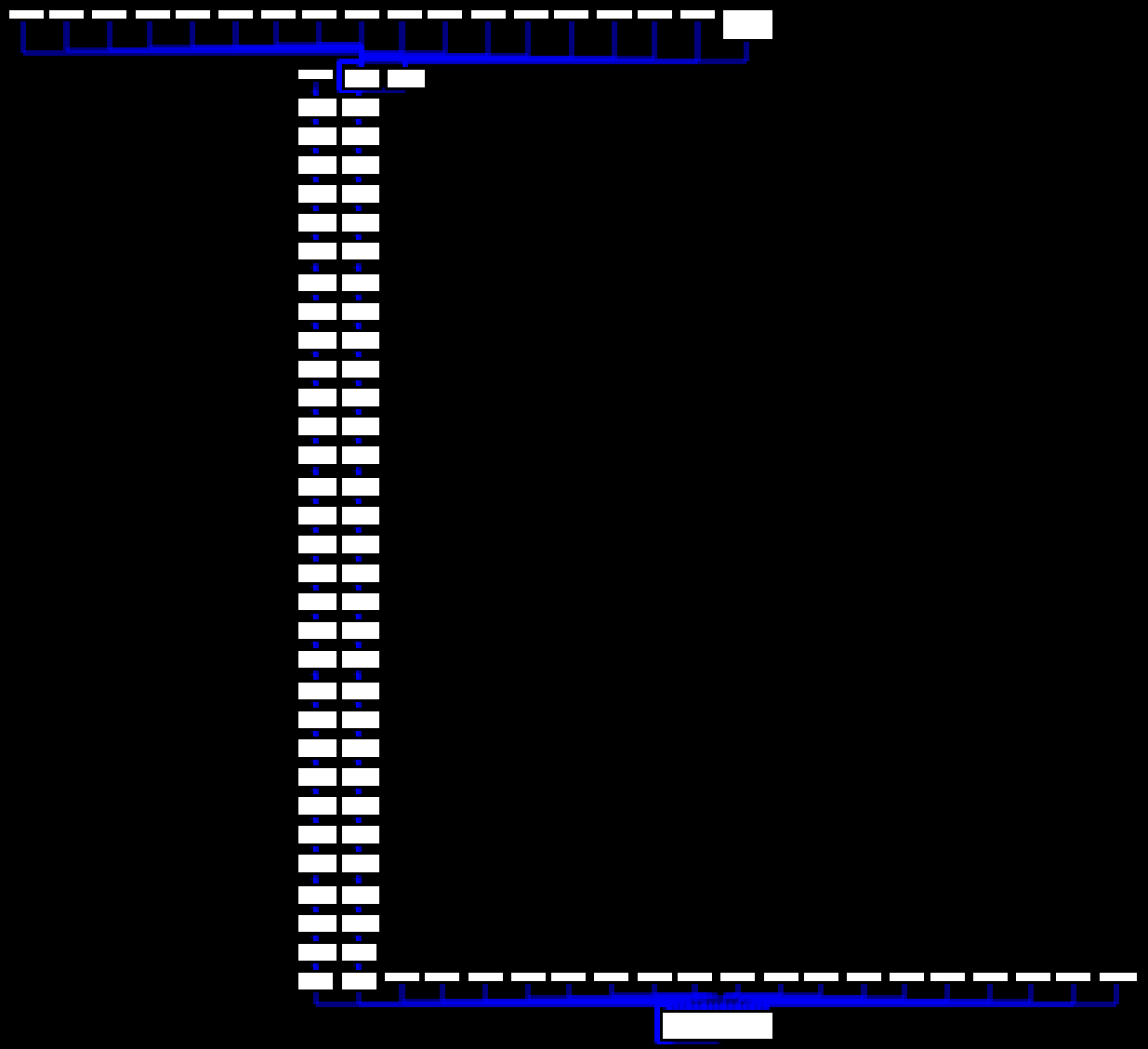


Observation

- IDA tries to align blocks in a given row

```
top:
jmp left
jmp top_end
… ; repeat
jmp right_side
top_end:
jmp $

left_side:
jmp $+2
… ; repeat
jmp bottom_left
```

```
right_side:
jmp $+2
… ; repeat
jmp bottom_right

bottom:
botton_left:
jmp bottom_end
… ; repeat
bottom_right:
bottom_end:
ret
```



Observation

- IDA tries to keep rows/columns together
  - But minimize branching distance

- Hour of tinkering
- Couldn’t make it work
- Try something else

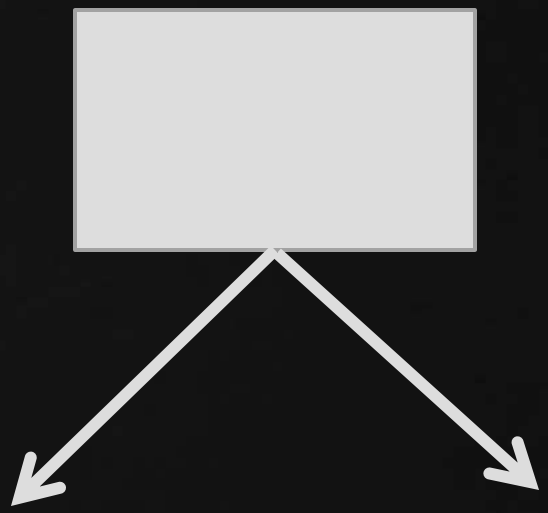
Separating the columns

- We have some control over how rows are arranged
  - Depends on nodes between
- IDA has all the control over columns
  - Can rearrange parent nodes and branches to keep columns close together

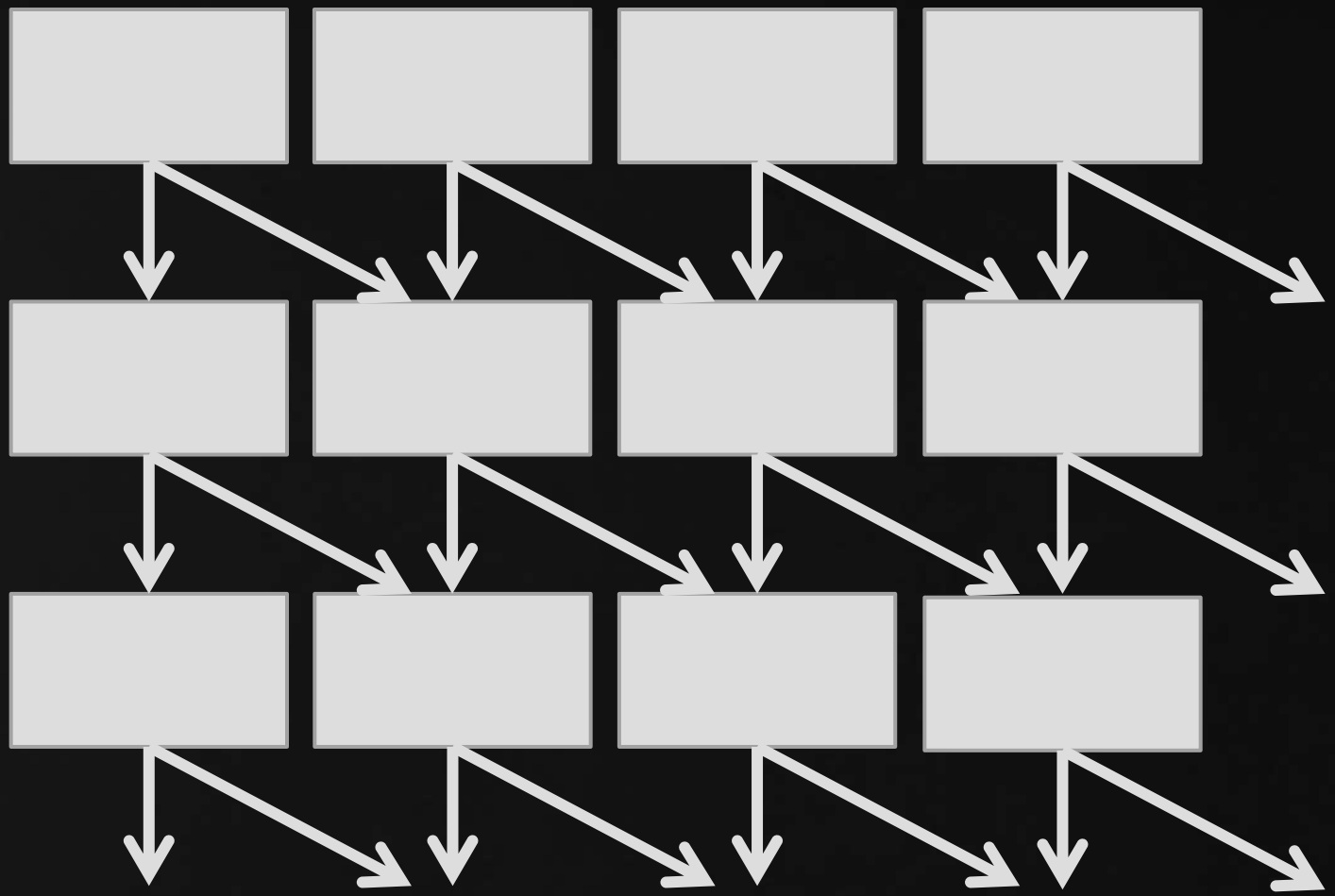
R.I.P. Idea 1

Idea 2

- *Force* IDA to keep things in order
  - Tie nodes together as tightly as possible
  - Prevent rearranging

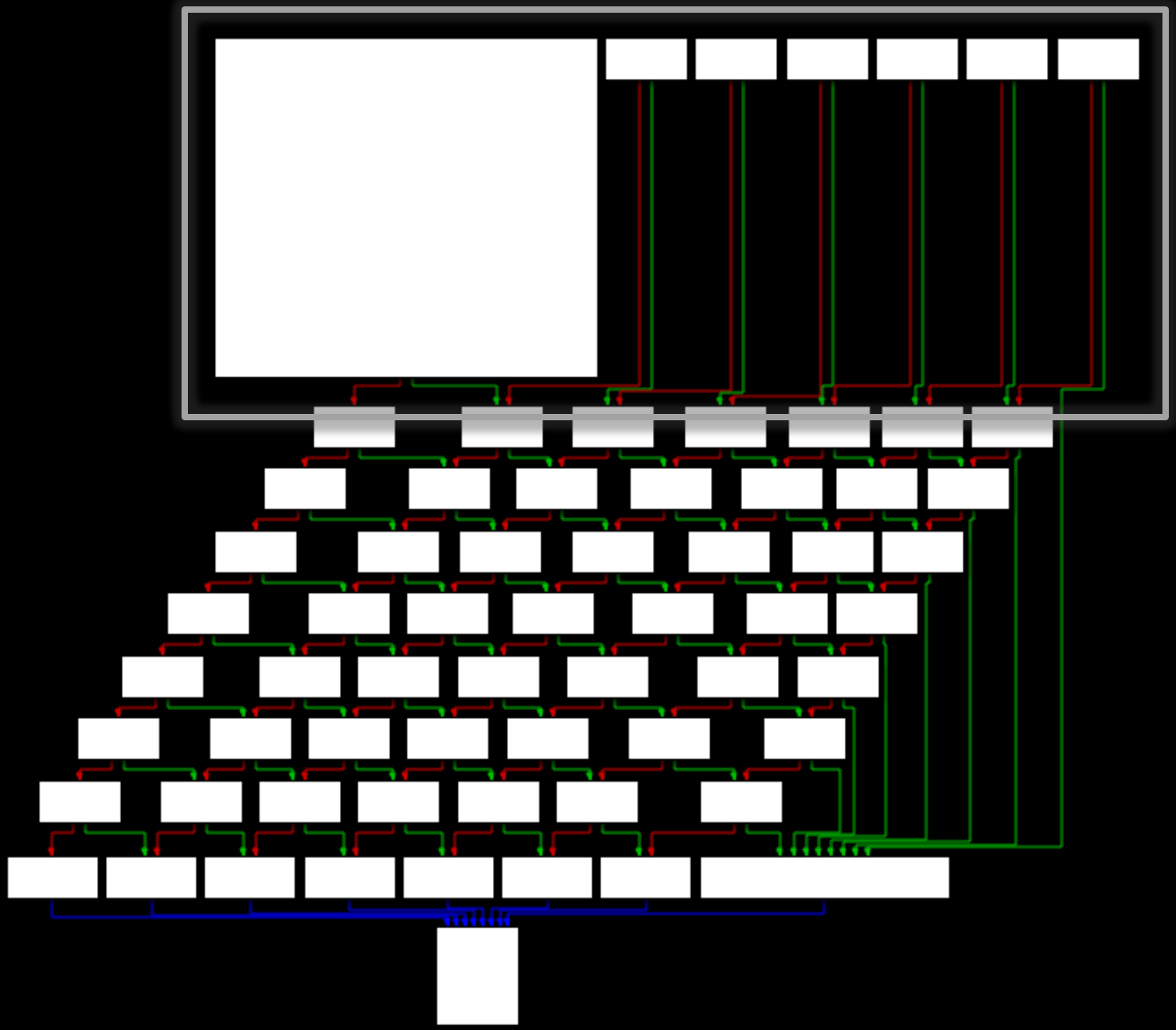


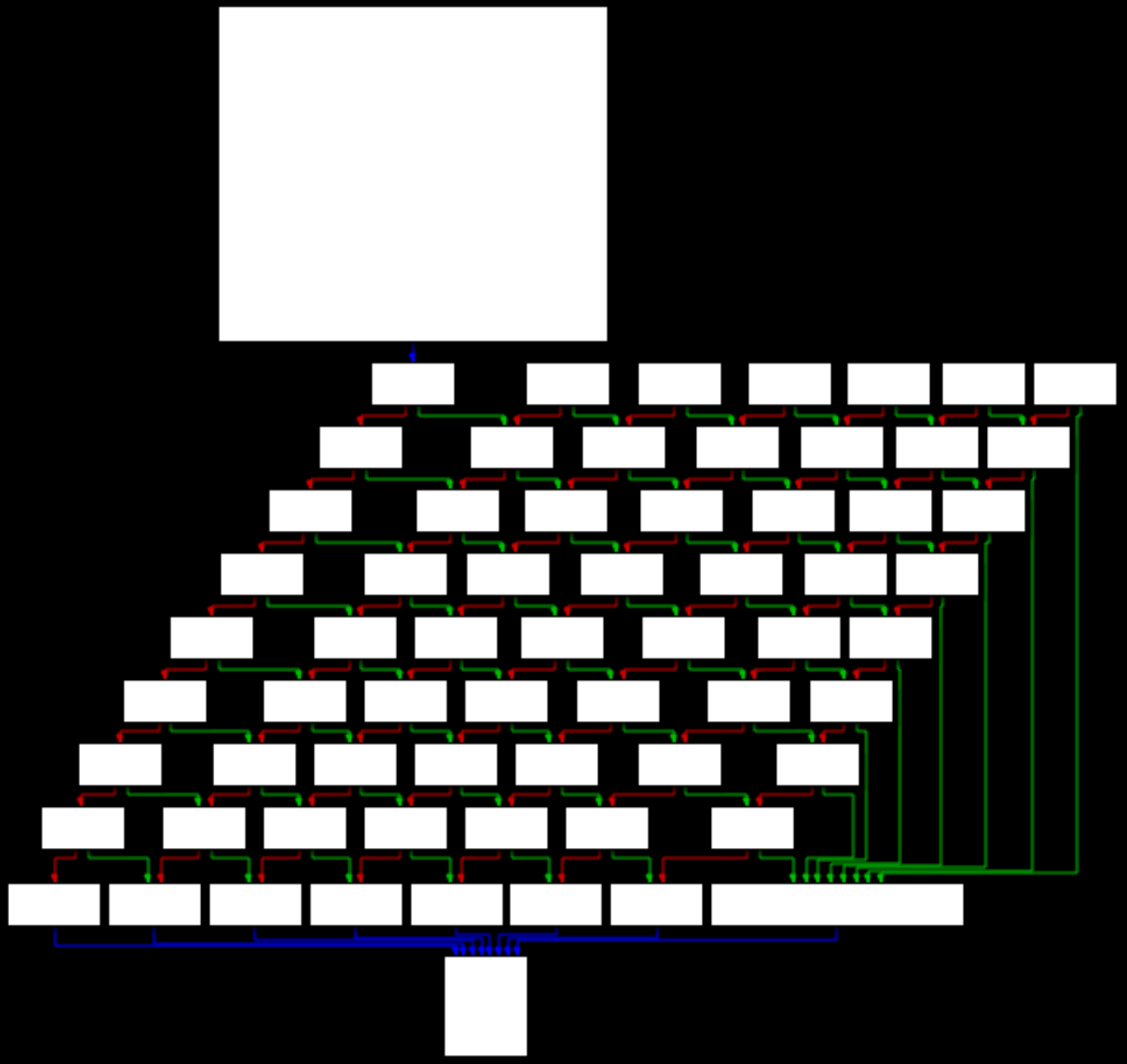
A node



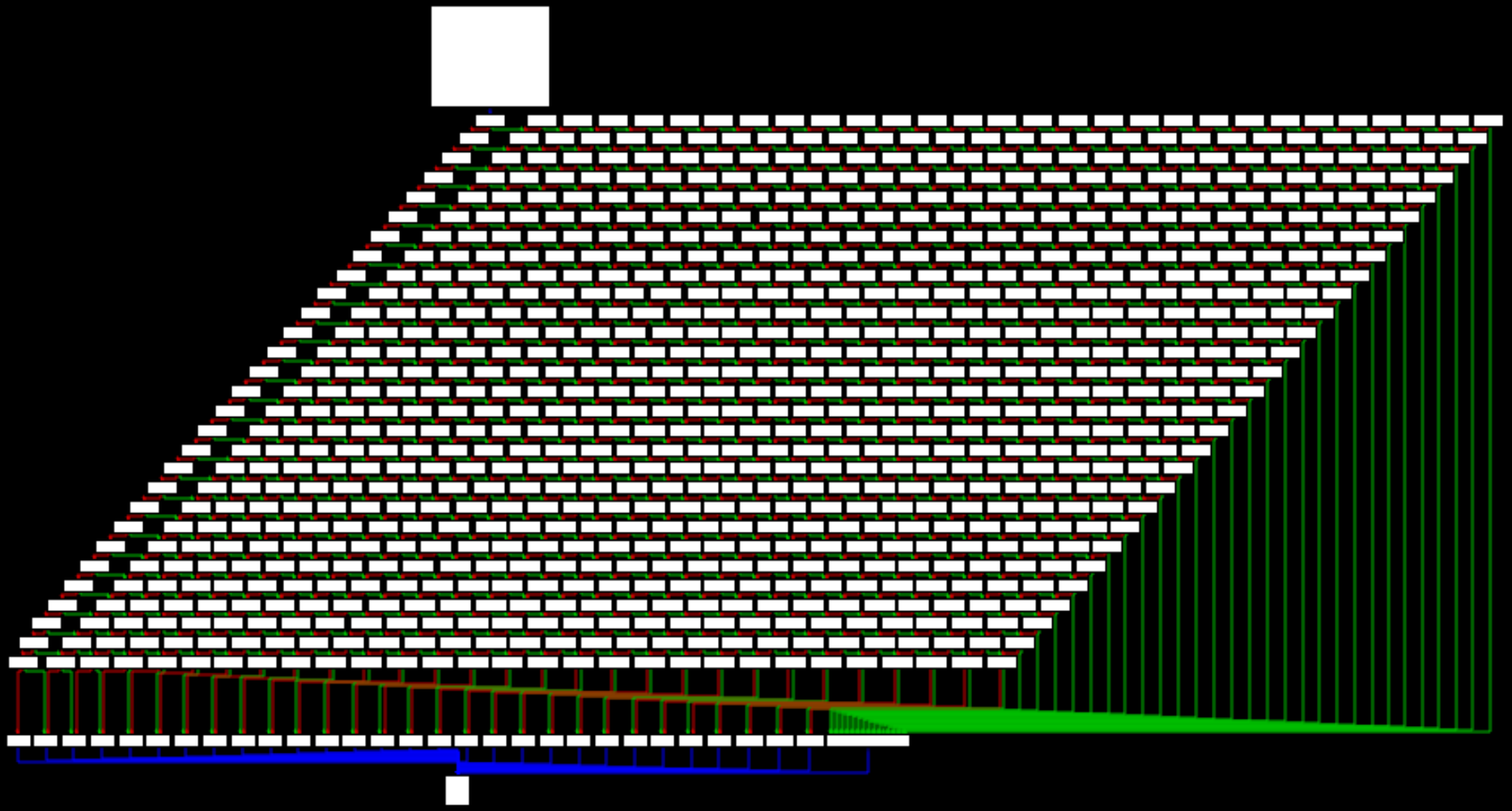
A tightly woven CFG

```
x:

a0: je b1     a1: je b2     a2: je b3     a3:
b0: je c1     b1: je c2     b2: je c3     b3:
c0: je d1     c1: je d2     c2: je d3     c3:
d0: jmp F     d1: jmp F     d2: jmp F     d3: jmp F

                                          F:
```



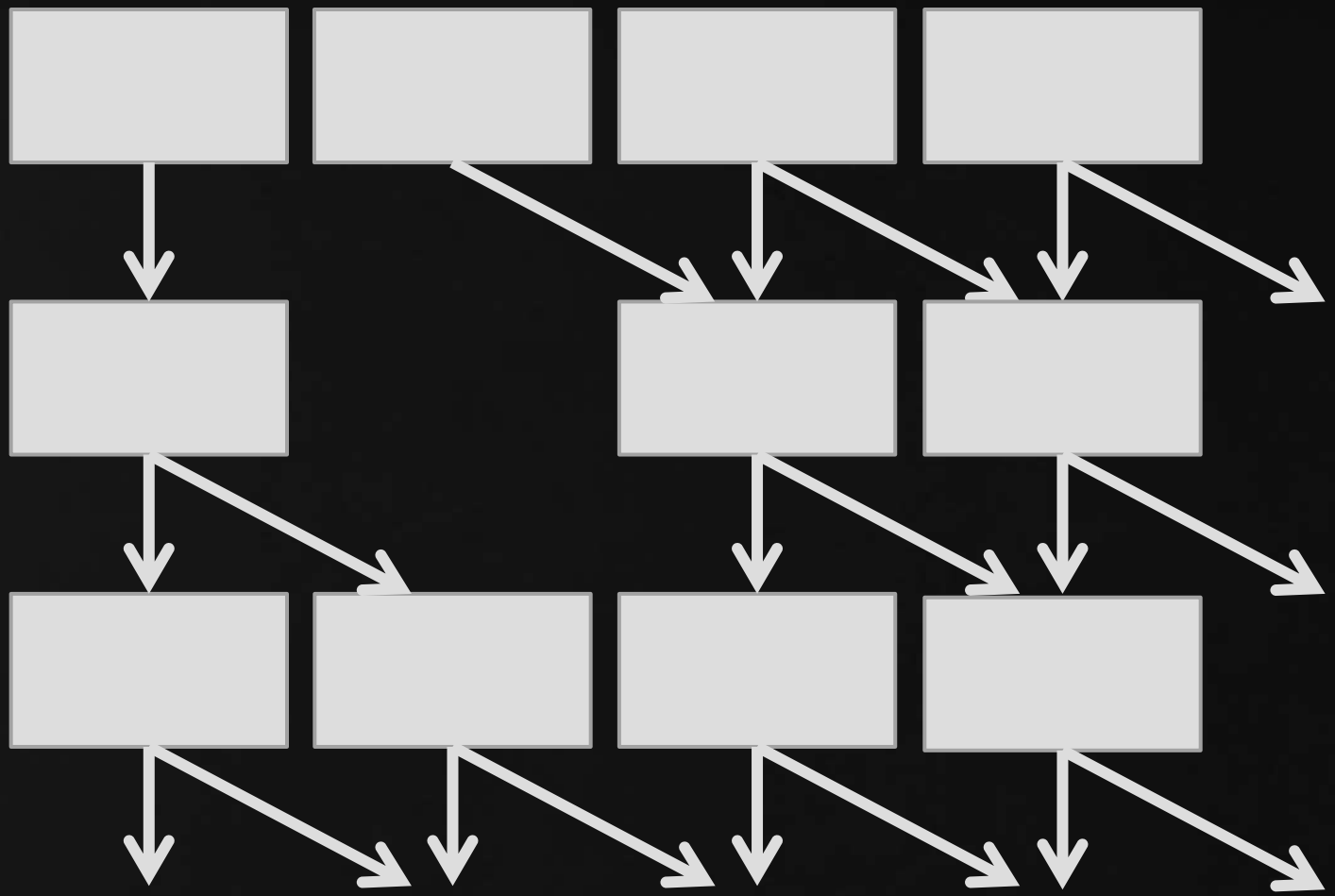


```
%macro column 3-4 "nonempty"
    %assign r 0
    %assign c %1
    %rep %2-1
        %assign nr r+1
        %assign nc c+1
        e_%+r%+_%+c:
        %ifidn %4, "empty"
        %else
            je e_%+nr%+_%+nc
        %endif
        %assign r r+1
    %endrep
    e_%+r%+_%+c: jmp %3
%endmacro
```

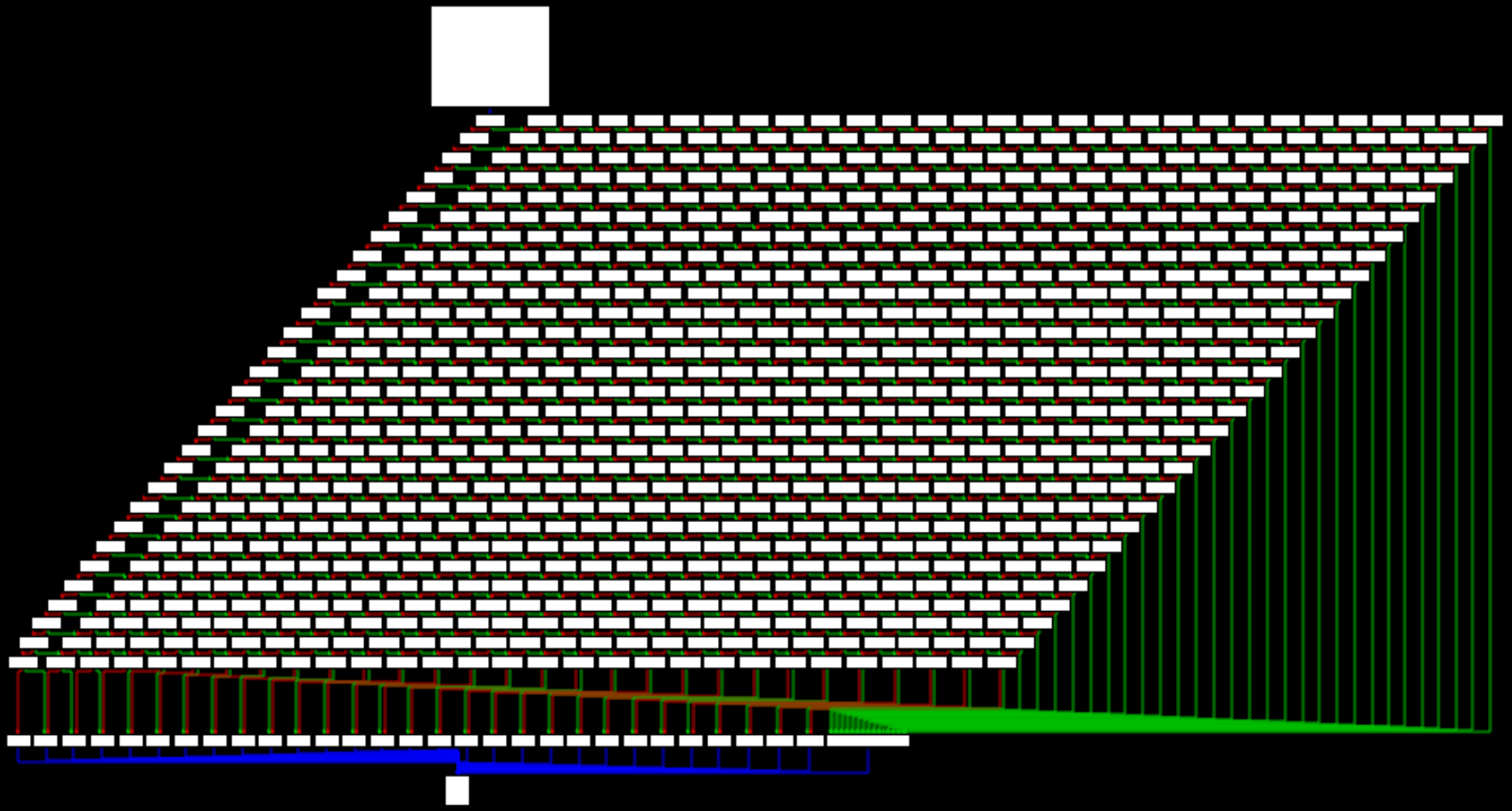


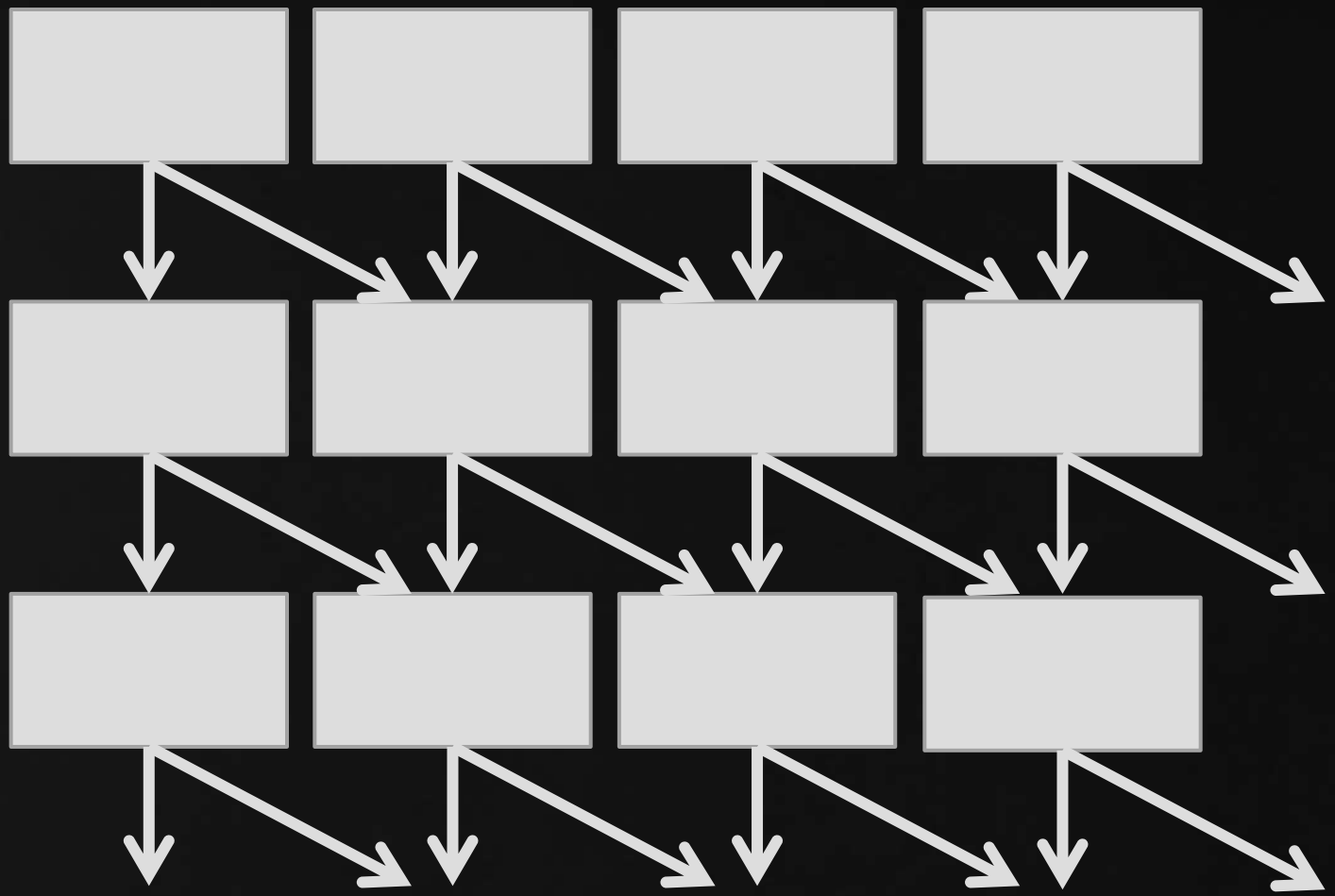
Idea 2, continued

- “Weave” the CFG together
- Turn “pixel” off by removing node?

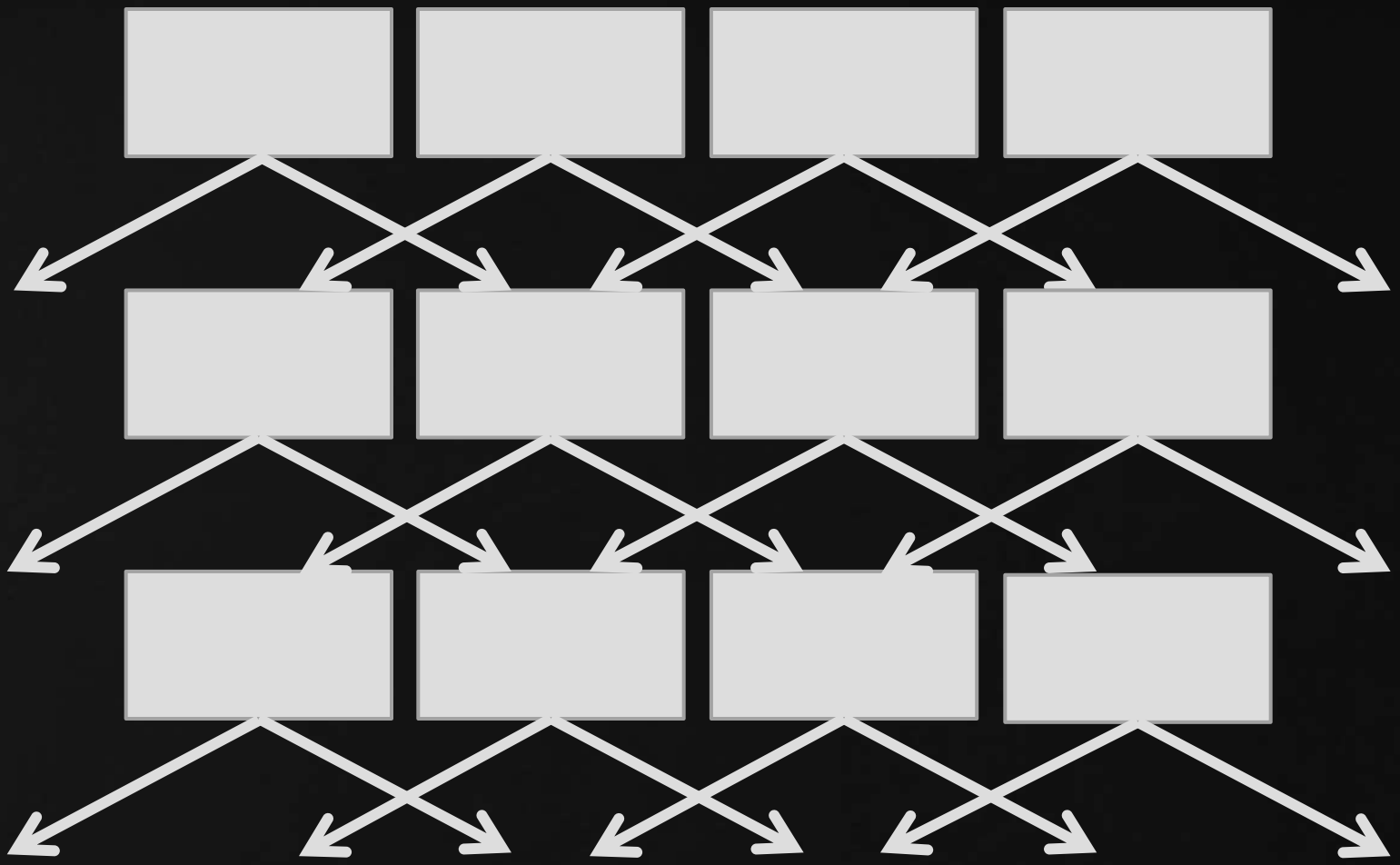


A tightly woven CFG





A tightly woven CFG

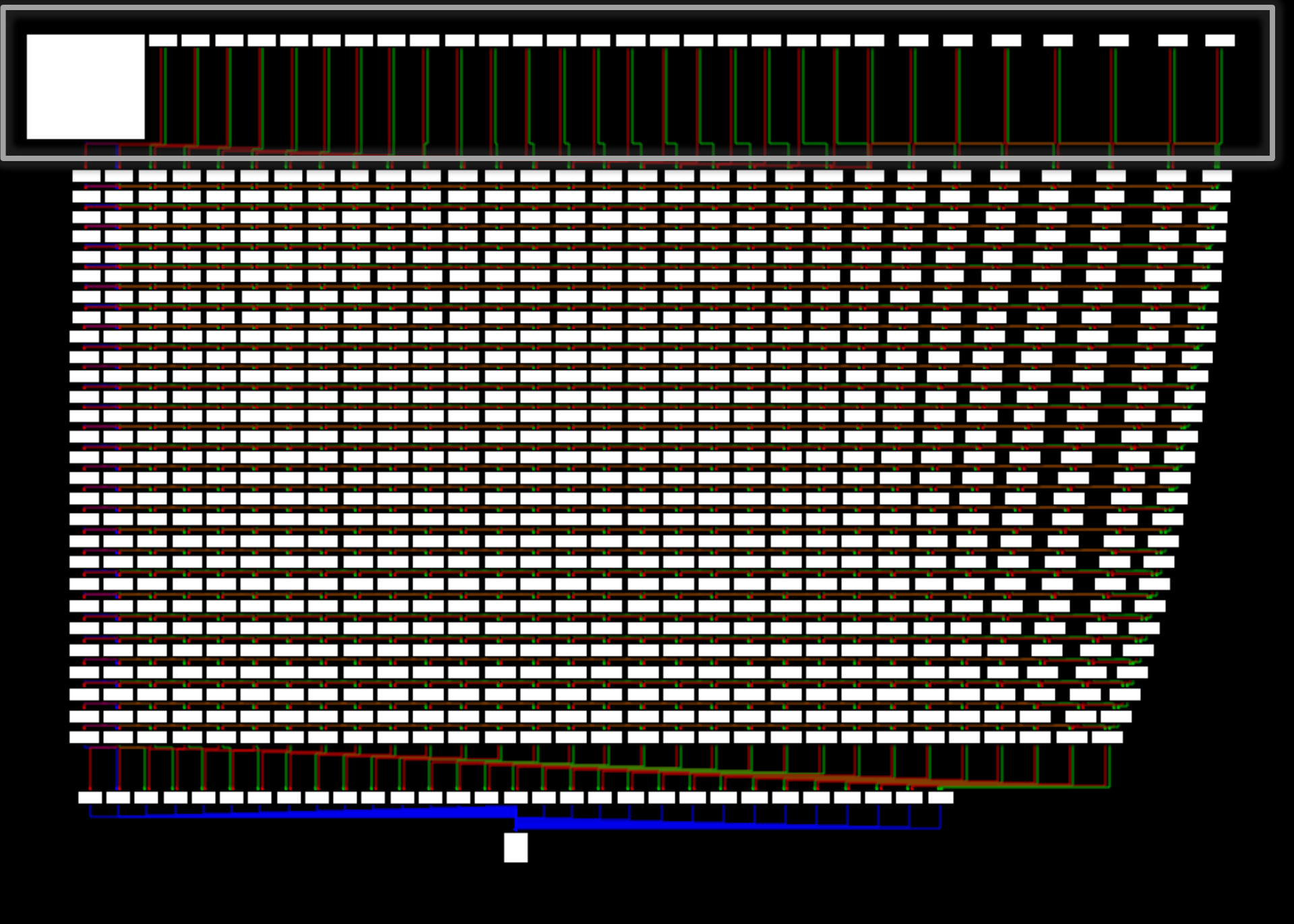


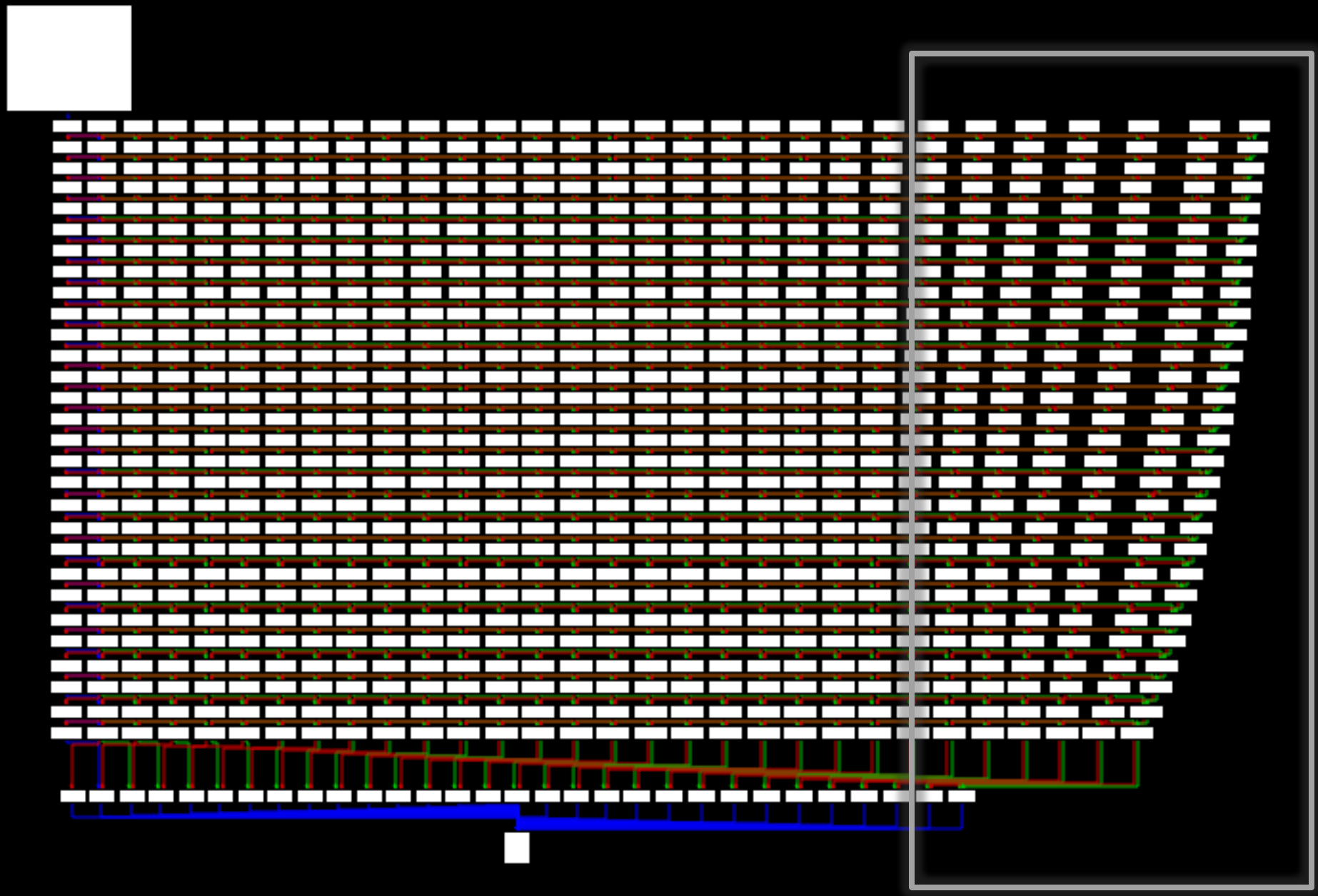
A tightly woven CFG, II

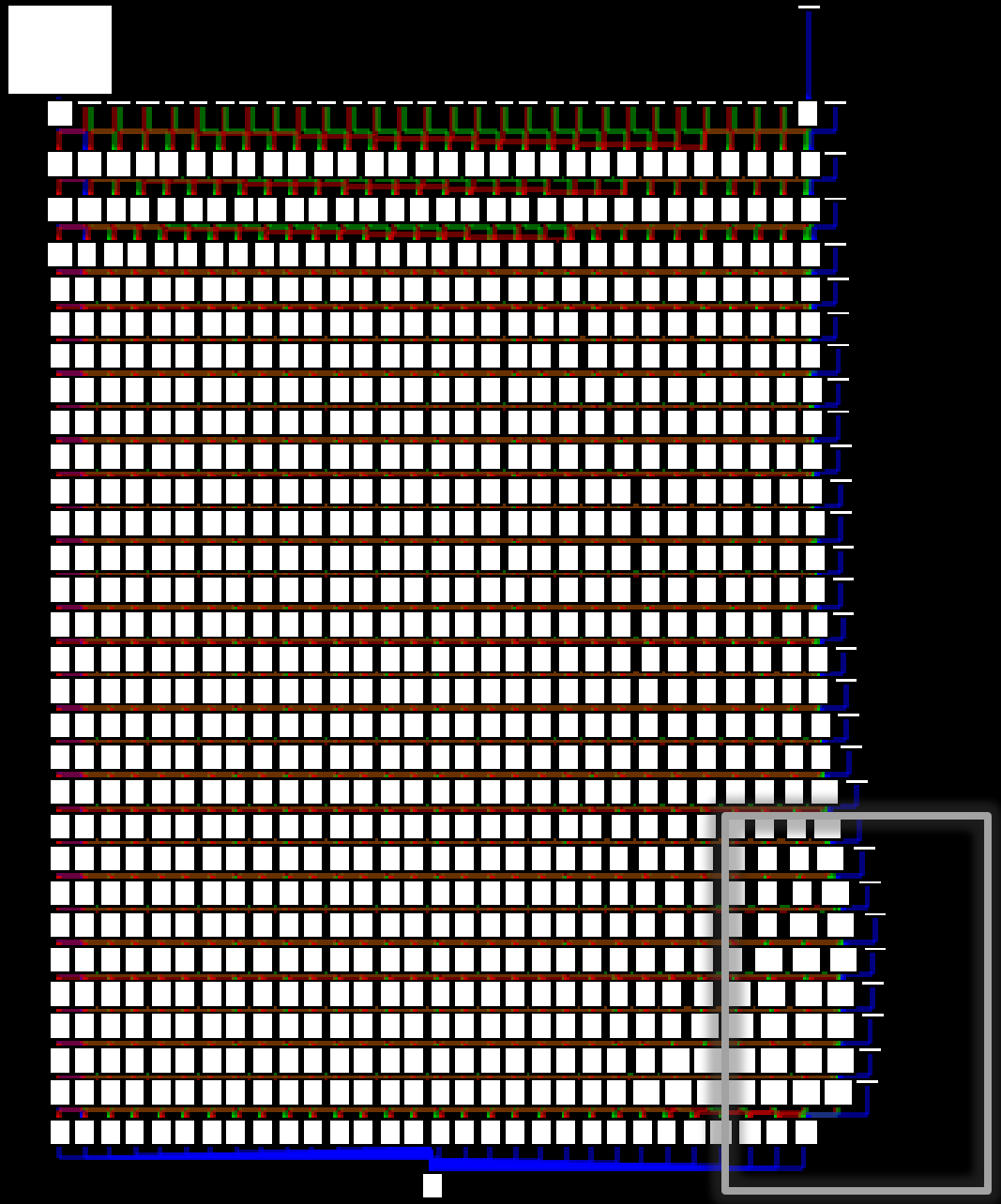
```
                          ; e_0_0 e_0_1 e_0_2 e_0_3
                          ; e_1_0 e_1_1 e_1_2 e_1_3
x:                        ; e_2_0 e_2_1 e_2_2 e_2_3
                          ; e_3_0 e_3_1 e_3_2 e_3_3
e_0_0:  je e_1_1
        jmp done

e_0_1:  je e_2_1  e_0_3:  je done
e_1_0:  je e_2_1  e_1_2:  je e_2_3  e_2_3:  je done
        jmp done  e_2_1:  je e_3_2  e_3_2:  jmp done
                  e_3_0:  jmp done
e_0_2:  je e_1_3                    e_3_3:  jmp done
e_1_1:  je e_2_2  e_1_3:  je done
e_2_0:  je e_3_1  e_2_2:  je e_3_3  done:
        jmp done  e_3_1:  jmp done  ret
```

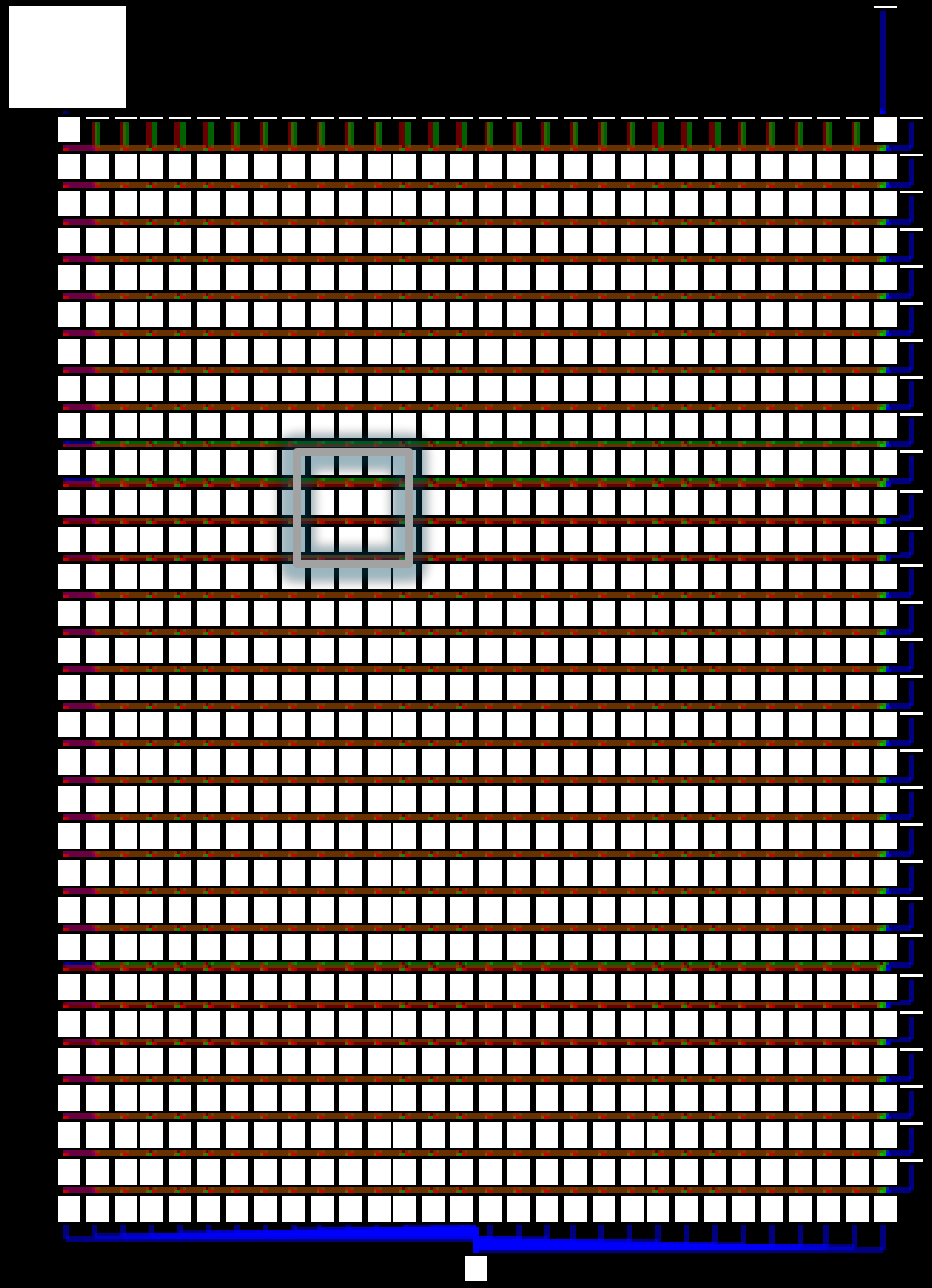
This file has been generated by The Interactive Disassembler (IDA)
Copyright (c) 2013 Hex-Rays, <support@hex-rays.com>
Chris Domas, Battelle Memorial Institute
; Format : ELF for Intel 386 (Relocatable)
; Source File : 'matrix_3.asm'
.model flat
_text segment para public 'CODE' use32
assume cs:_text
assume es:nothing, ss:nothing, ds:_text, fs:nothing, gs:nothing
public x
x proc near
jz short e_1_1 ; Alternative name is 'x'
; e_0_0
e_0_1:
jz short e_2_1
e_0_2:
jz short e_1_2
e_0_3:
jz short done
e_1_0:
jz short e_2_1
e_1_1:
jz short e_2_2
e_1_2:
jz short done
e_1_3:
jz short e_2_3
e_2_0:
jz short e_3_1
e_2_1:
jz short e_3_2
e_2_2:
jz short e_3_3
e_2_3:
jz short done
jmp short done
jmp short done
jmp short done
e_3_0:
jmp short done
e_3_1:
jmp short done
e_3_3:
jmp short $+2
e_3_2:
jmp short done
done:
retn
x endp
_text ends
end

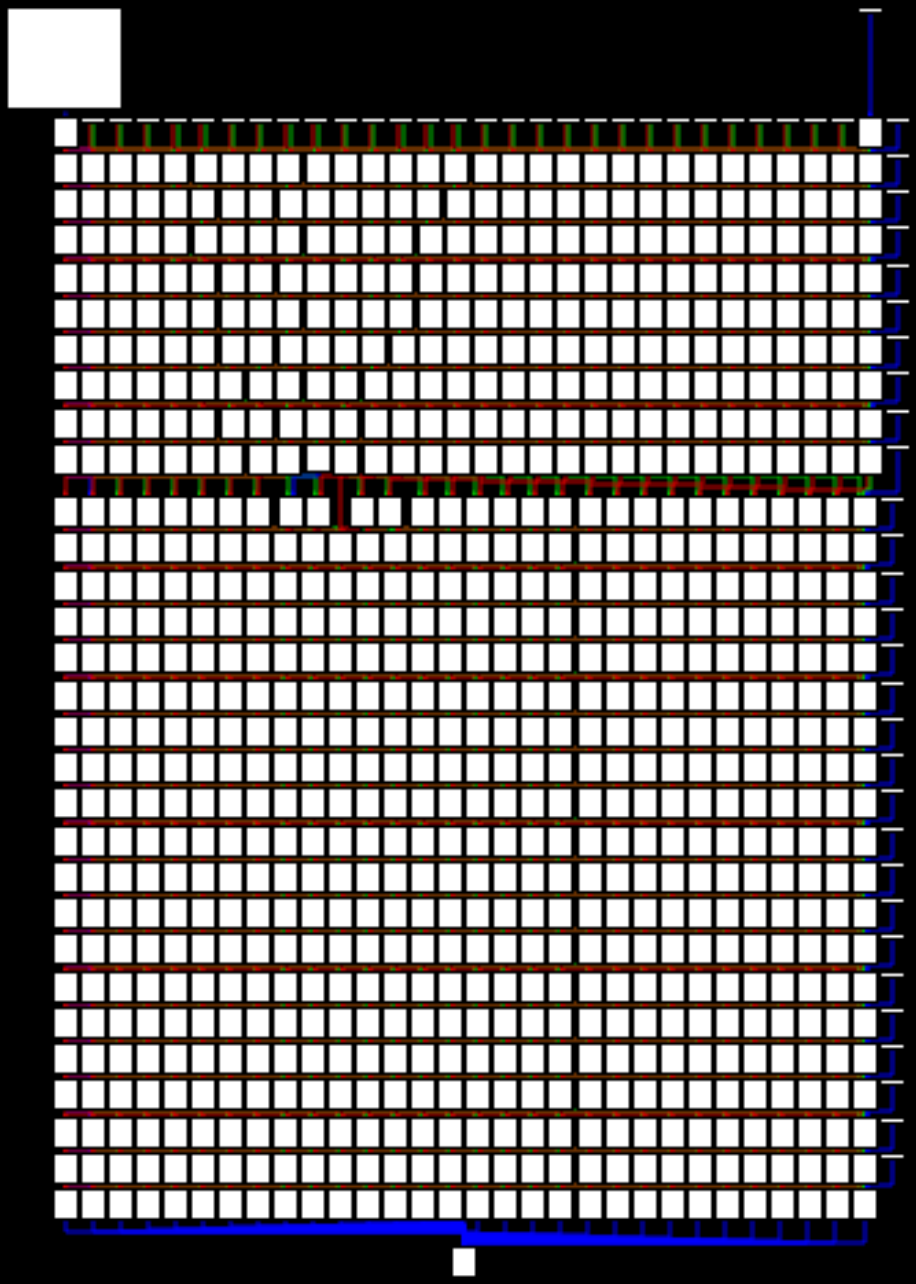
```
; row, column, width, height, done
%macro diag 5
    %assign r %1
    %assign c %2
    %assign width %3
    %assign height %4

    %rep 256 ; max size
        %assign nr r+1
        %assign nc c+1

        e_%+r%+_%+c:
        %if nr >= height
        %elif nc >= width
            je e_%+nr%+_%+c
        %else
            %if c == 0
                jmp e_%+nr%+_%+nc
                %exitrep
            %else
                je e_%+nr%+_%+nc
            %endif
        %endif

        %assign r r+1
        %assign c c-1
        %if r>=width
            jmp %5
            %exitrep
        %endif
    %endrep

%endmacro
```











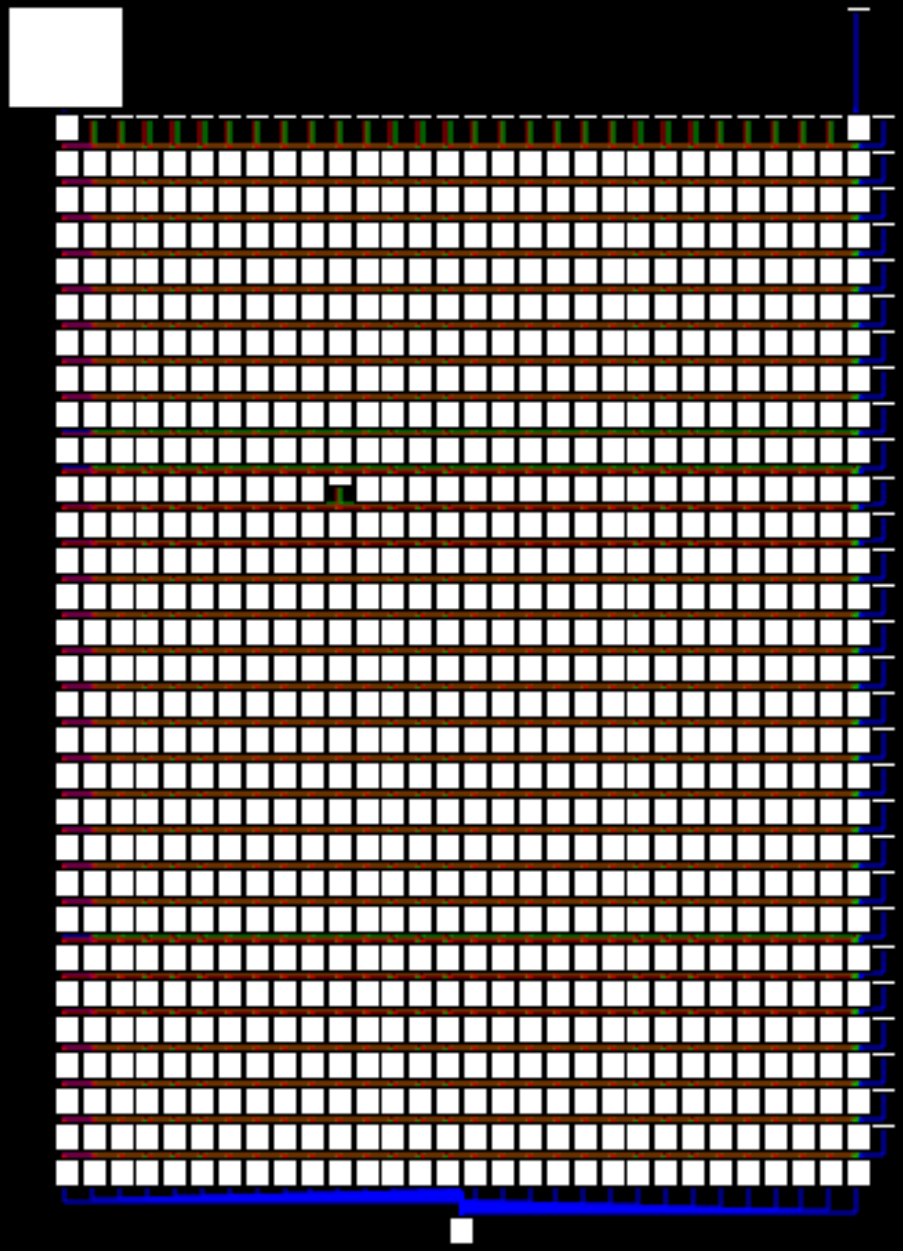
- We still can’t remove a node

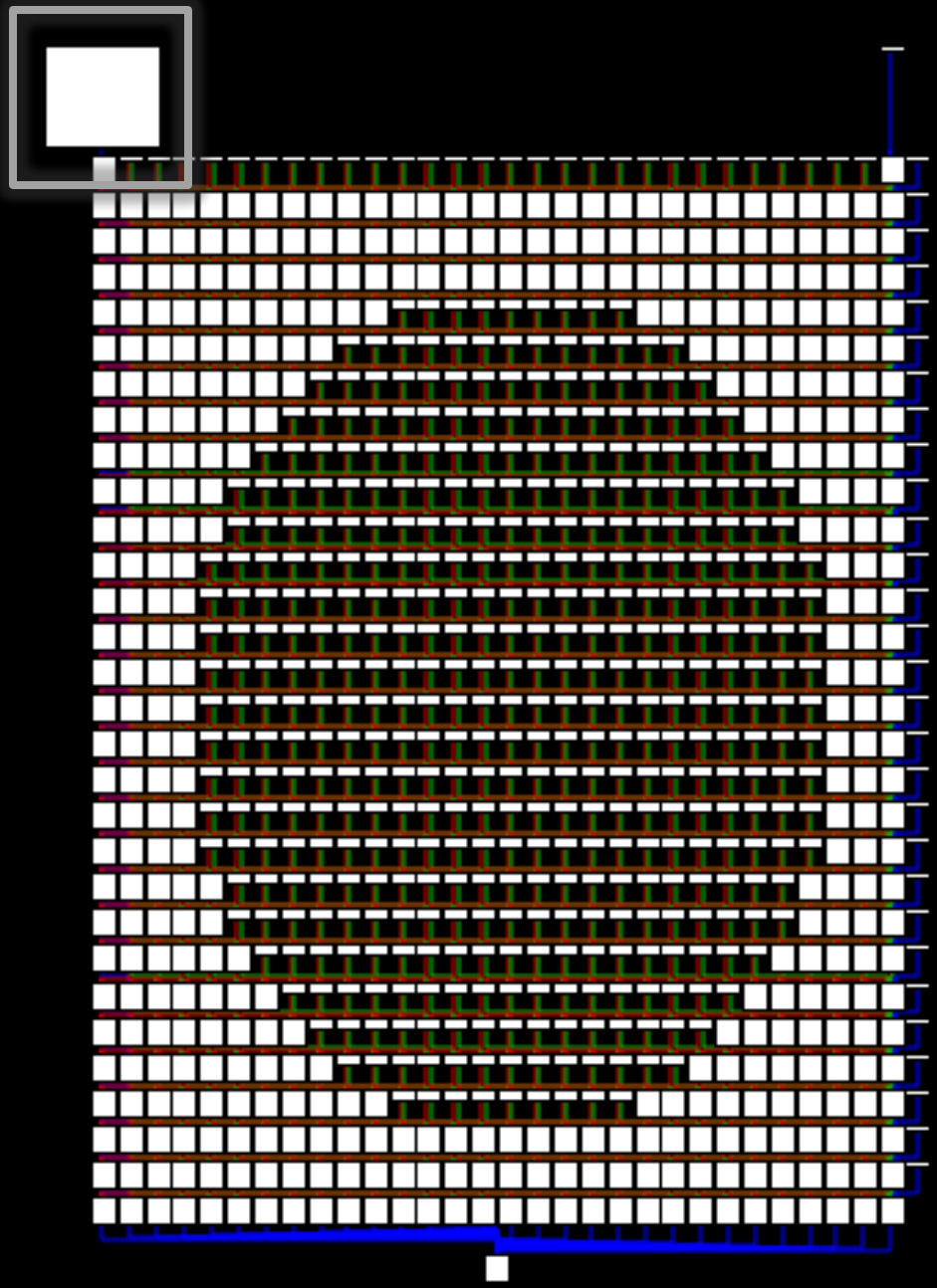
R.I.P. Idea 2

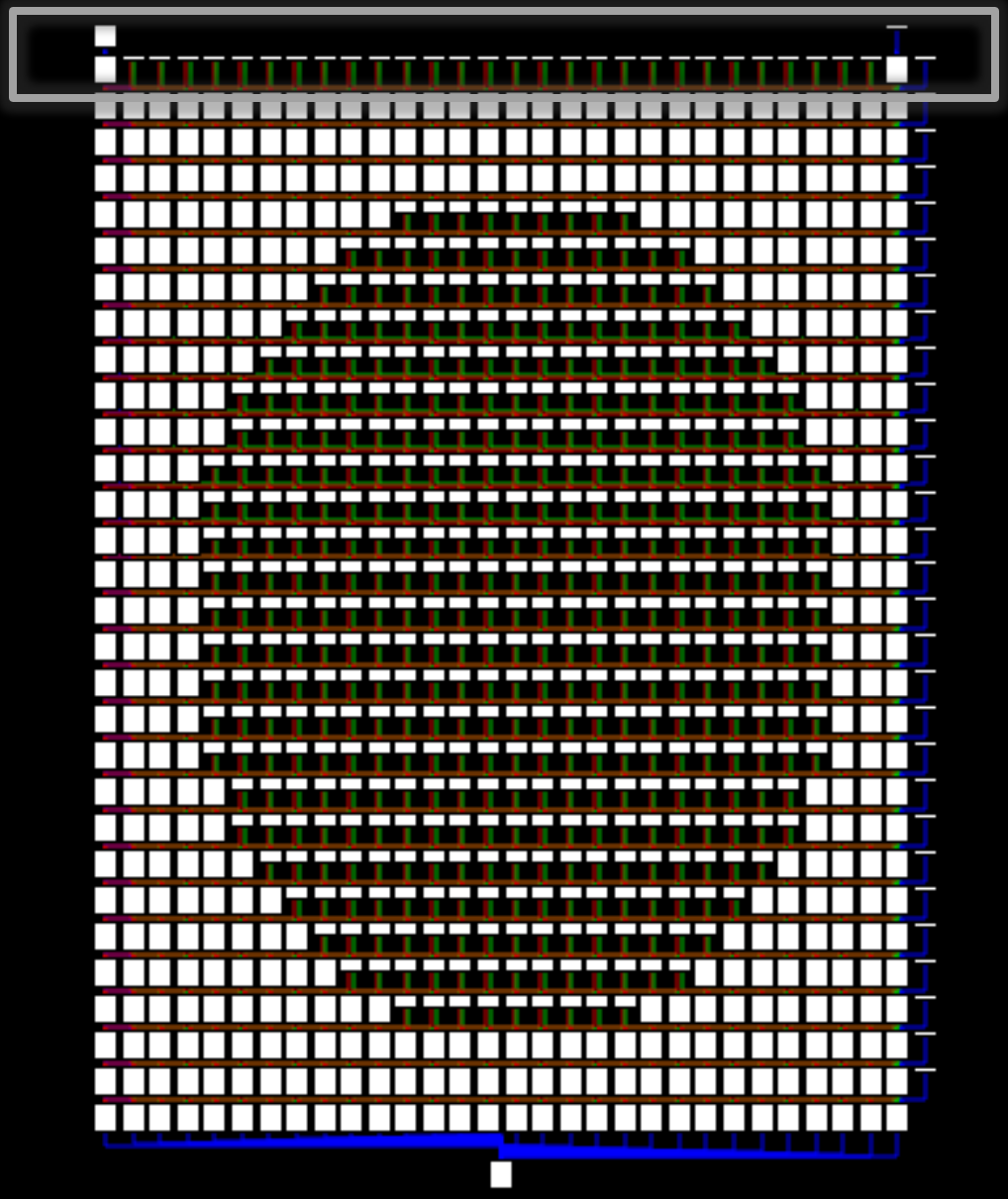
Idea 3

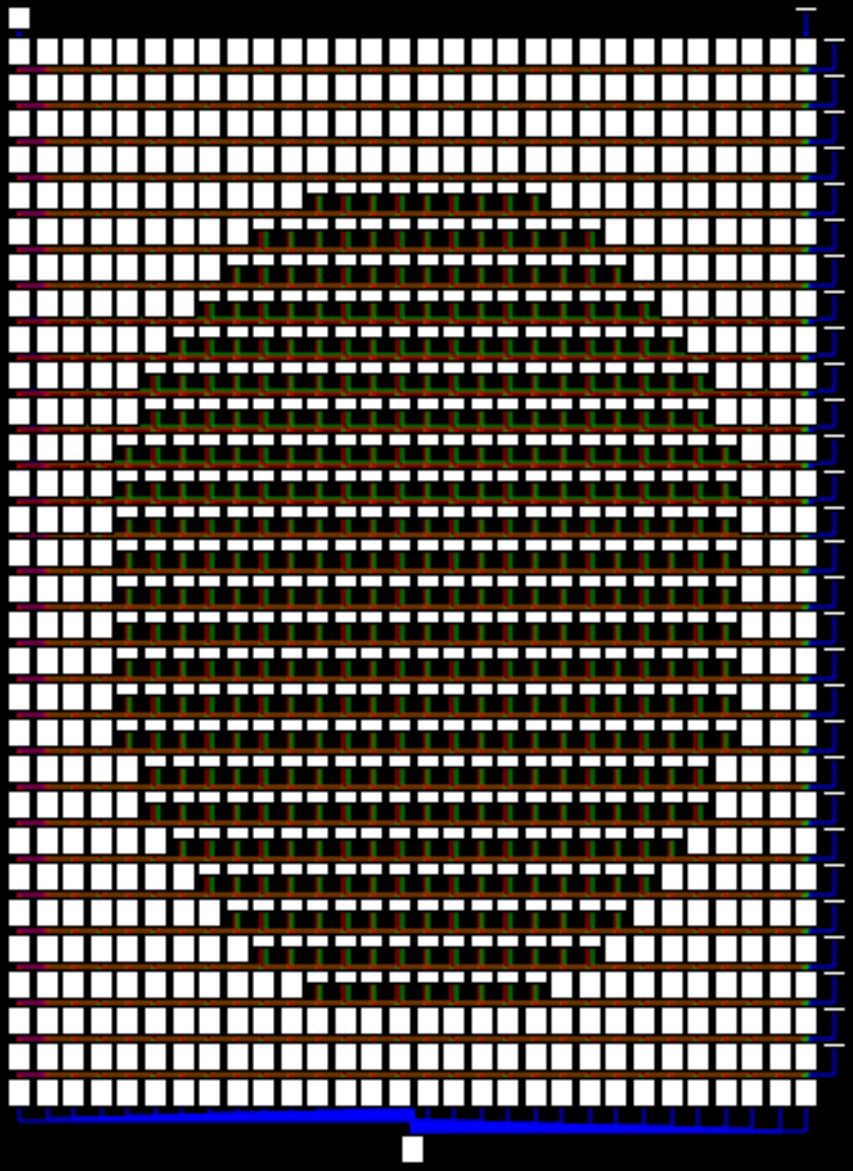
- Leave all nodes
- Fill with code if “on”
- Leave empty if “off”


e_9_8
jz e_9_9
jz e_9_10
jz e_9_11
jz e_9_12
jz e_9_13
jz
e_9_8:
nop
nop
nop
nop
nop
nop
jz e_9_9
e_9_9:
nop
nop
nop
nop
nop
nop
jz e_9_10
e_9_10:
nop
nop
nop
nop
nop
nop
jz e_9_11
e_9_11:
nop
nop
nop
nop
nop
nop
jz e_9_12
e_9_12:
nop
nop
nop
nop
nop
nop
jz e_9_13
e_9_13:
nop
nop
nop
nop
nop
nop
jz
e_10_8
e_10_8:
nop
nop
nop
nop
nop
nop
jz e_10_9
e_10_9:
nop
nop
nop
nop
nop
nop
jz e_10_10
e_10_10:
jz e_11_11
e_10_11:
nop
nop
nop
nop
nop
nop
jz e_10_12
e_10_12:
nop
nop
nop
nop
nop
nop
jz e_10_13
e_10_13:
nop
nop
nop
nop
nop
nop
jz
e_11_8
e_11_8:
nop
nop
nop
nop
nop
nop
jz e_11_9
e_11_9:
nop
nop
nop
nop
nop
nop
jz e_11_10
e_11_10:
nop
nop
nop
nop
nop
nop
jz e_11_11
e_11_11:
nop
nop
nop
nop
nop
nop
jz e_11_12
e_11_12:
nop
nop
nop
nop
nop
nop
jz e_11_13
e_11_13:
nop
nop
nop
nop
nop
nop
jz
e_12_8
e_12_8:
nop
nop
nop
nop
nop
nop
jz e_12_9
e_12_9:
nop
nop
nop
nop
nop
nop
jz e_12_10
e_12_10:
nop
nop
nop
nop
nop
nop
jz e_12_11
e_12_11:
nop
nop
nop
nop
nop
nop
jz e_12_12
e_12_12:
nop
nop
nop
nop
nop
nop
jz e_12_13
e_12_13:
nop
nop
nop
nop
nop
nop
jz



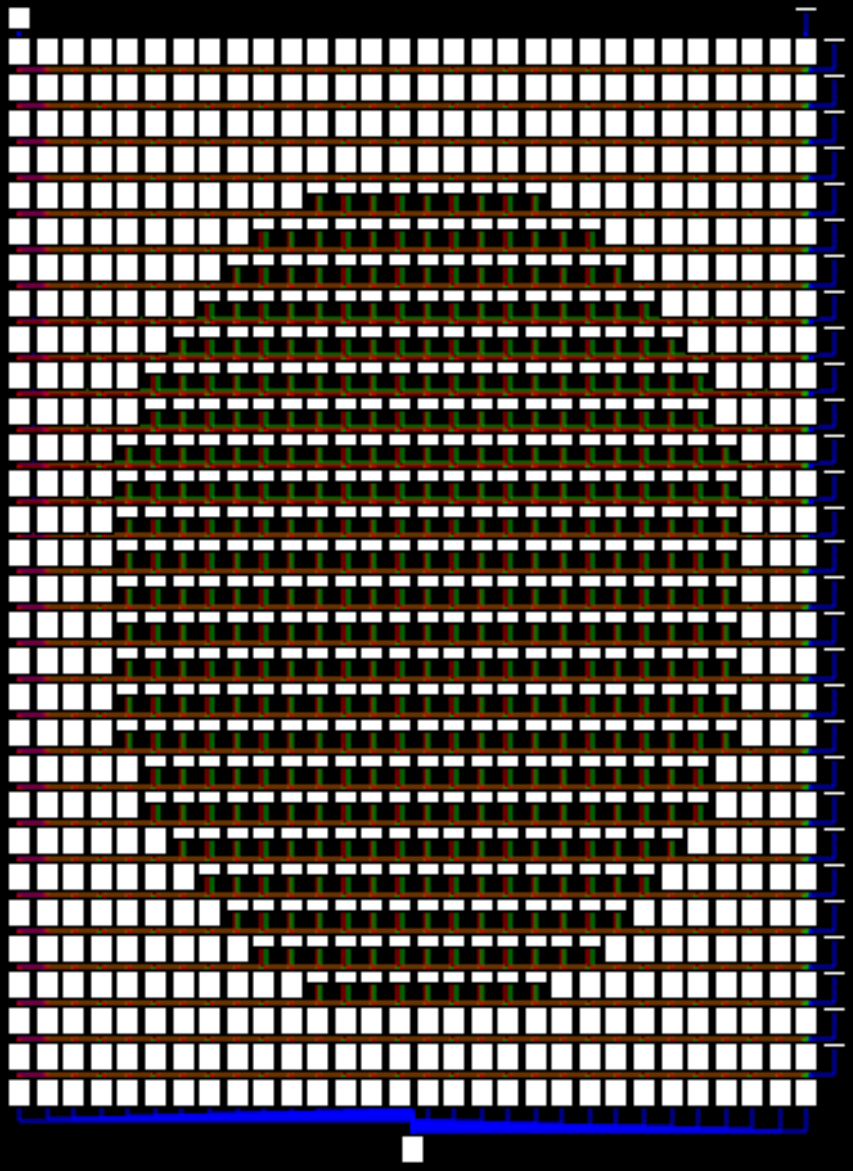


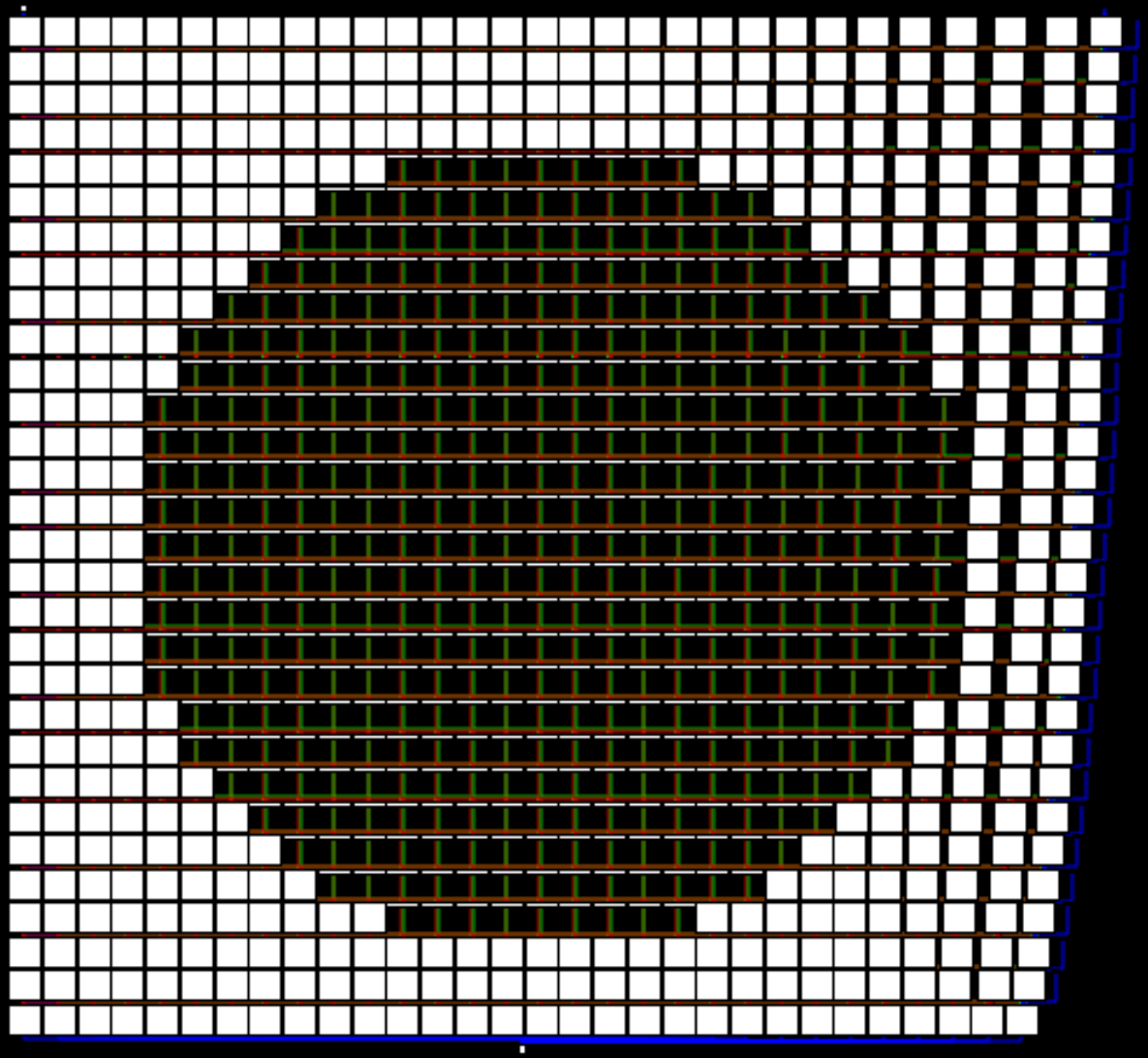


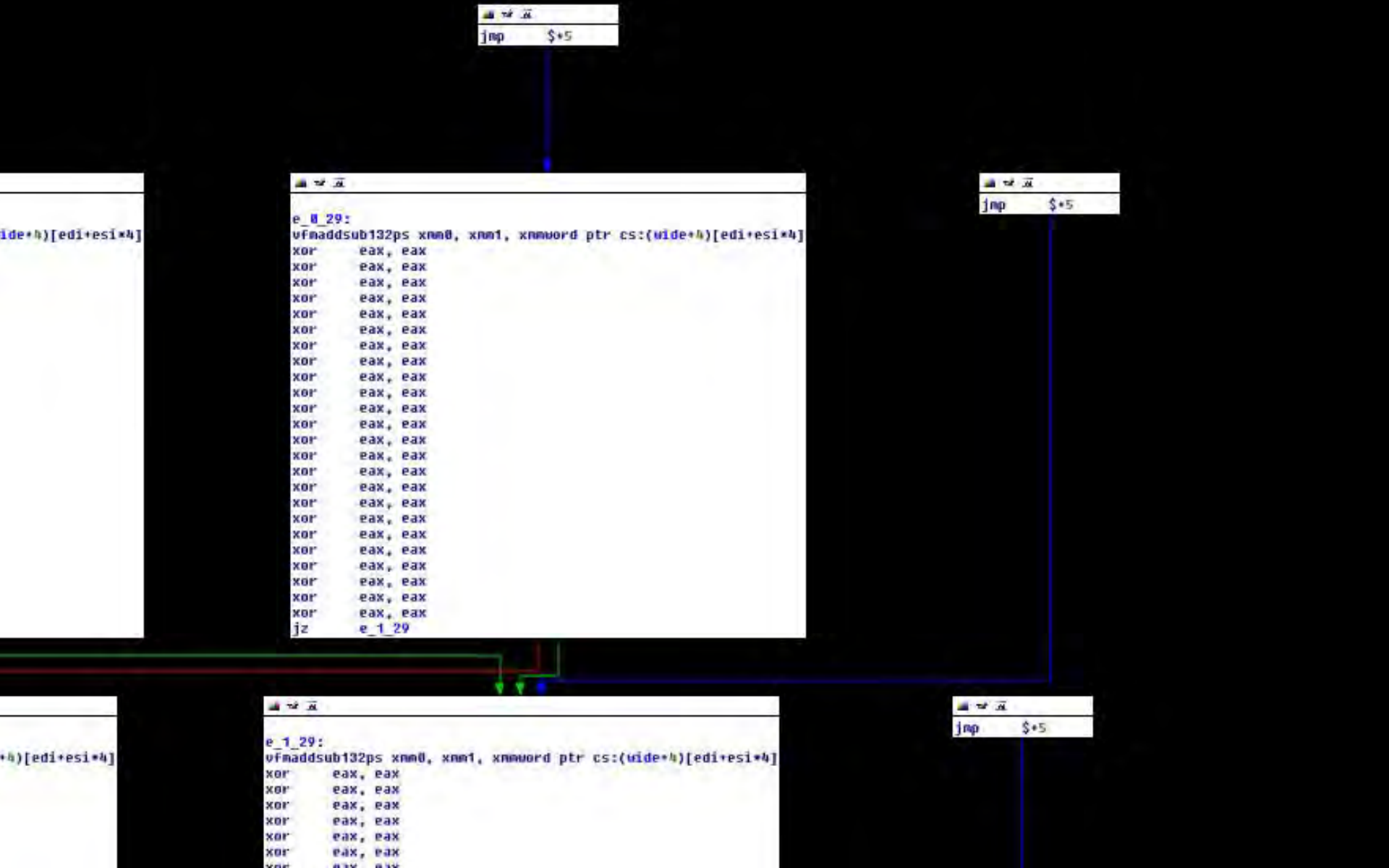


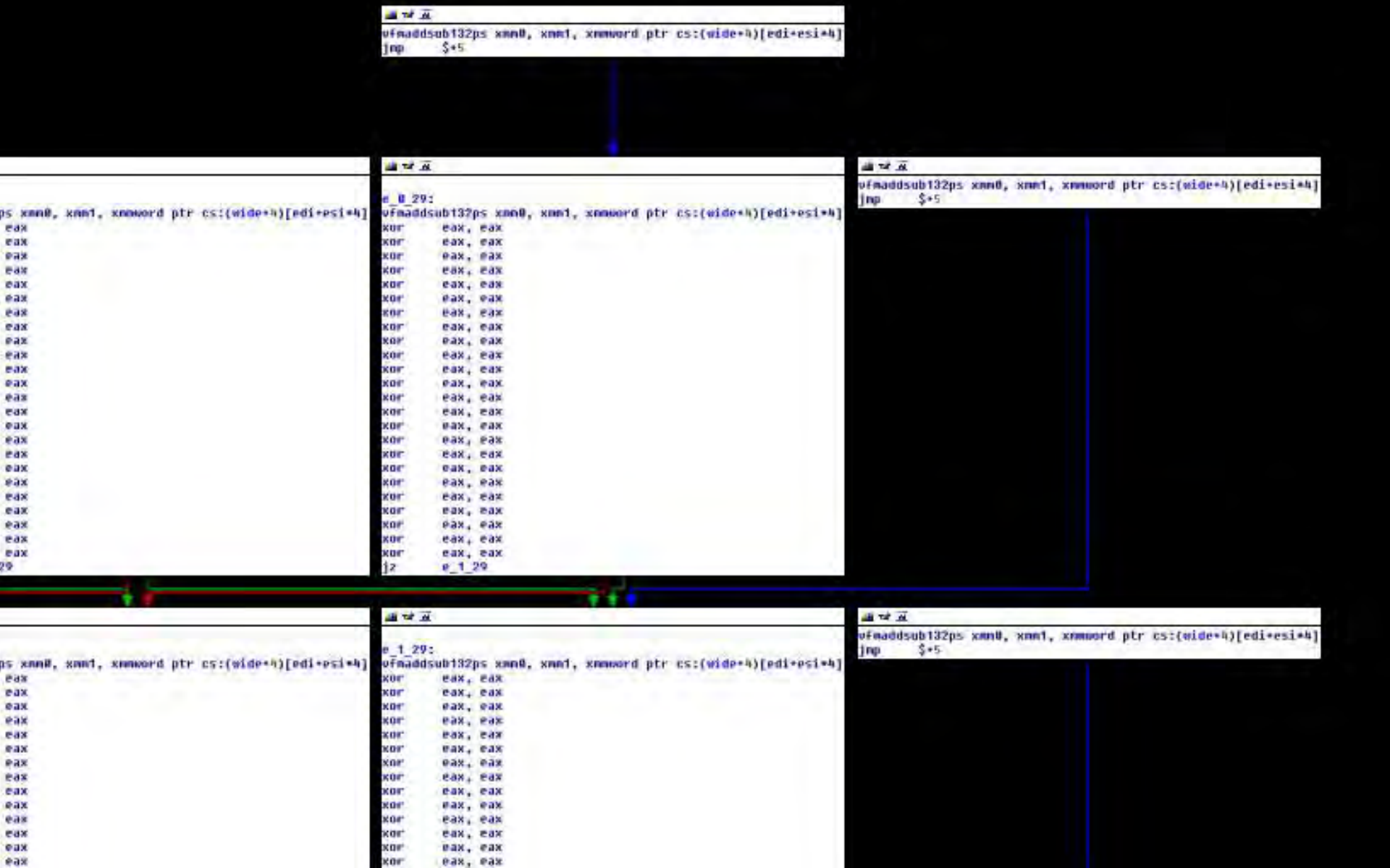
Enhance contrast

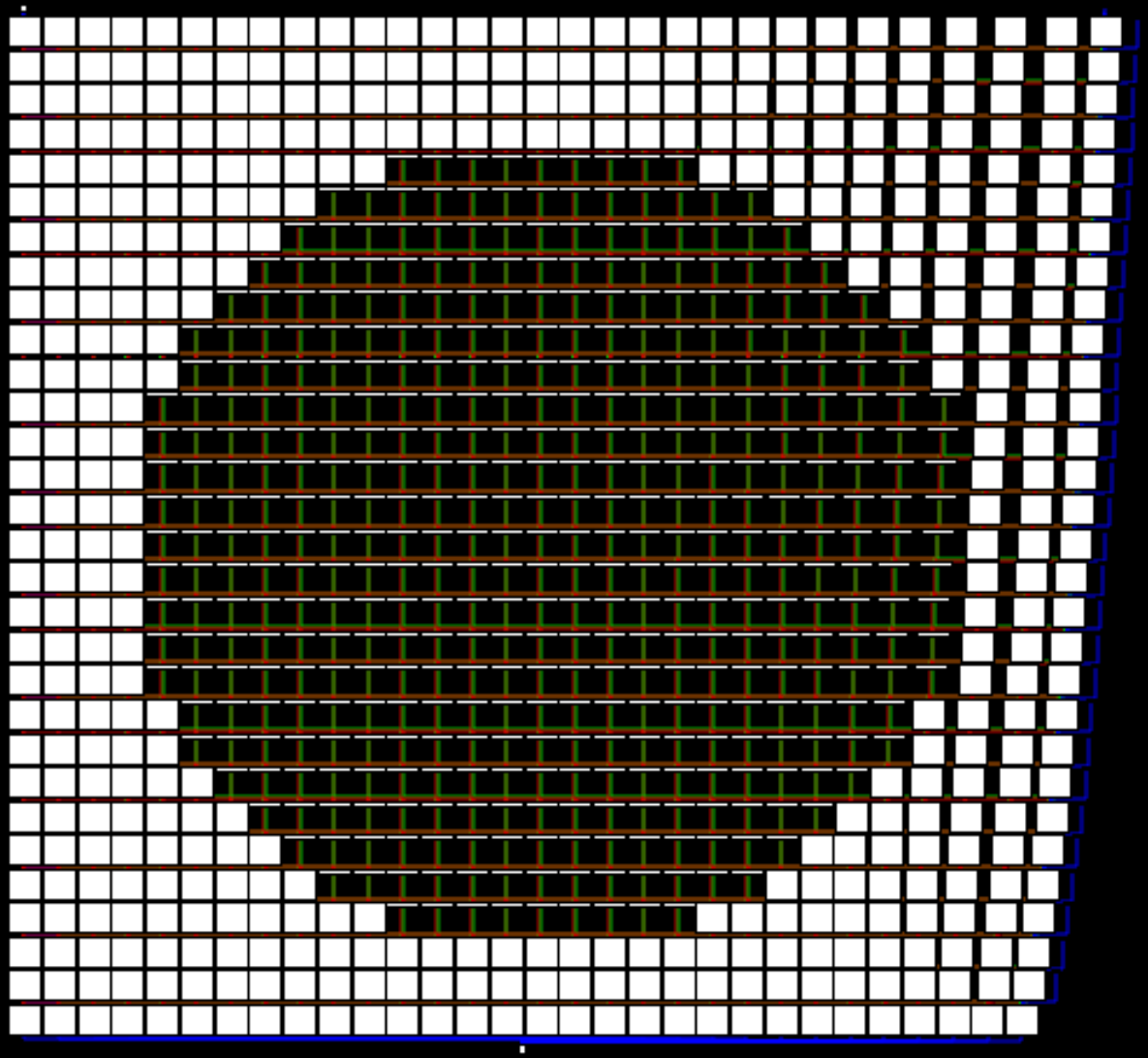
- “Empty” pixel still needs 2 lines
- Increase contrast by reducing impact of those 2
- Reduce impact by increasing height
- Increase height by increasing width
- vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8068860h]

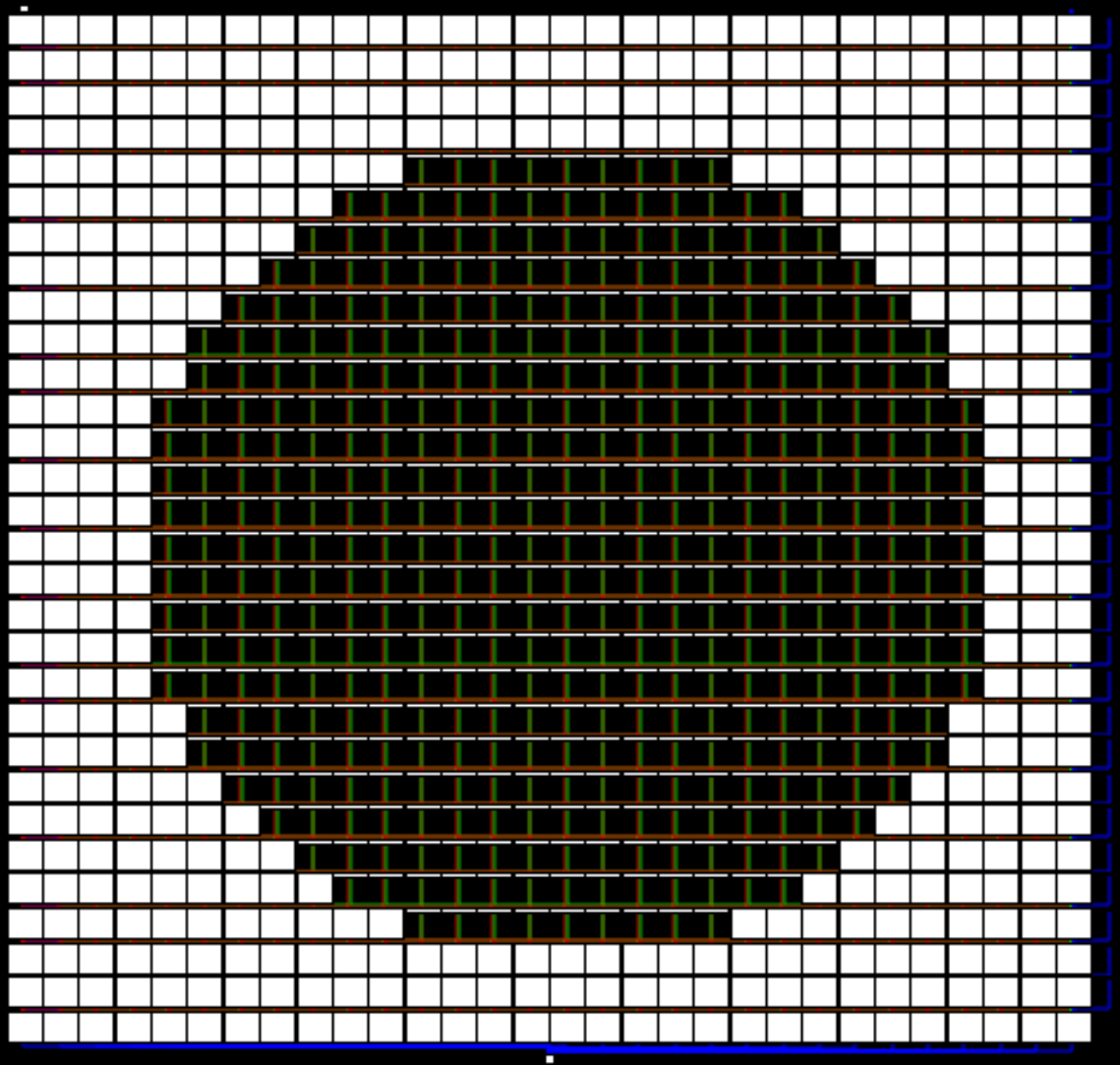





jmp $+5
e_0_29:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:(wide+4)[edi+esi*4]
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
jz e_1_29
jmp $+5
e_1_29:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:(wide+4)[edi+esi*4]
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
xor eax, eax
jmp $+5

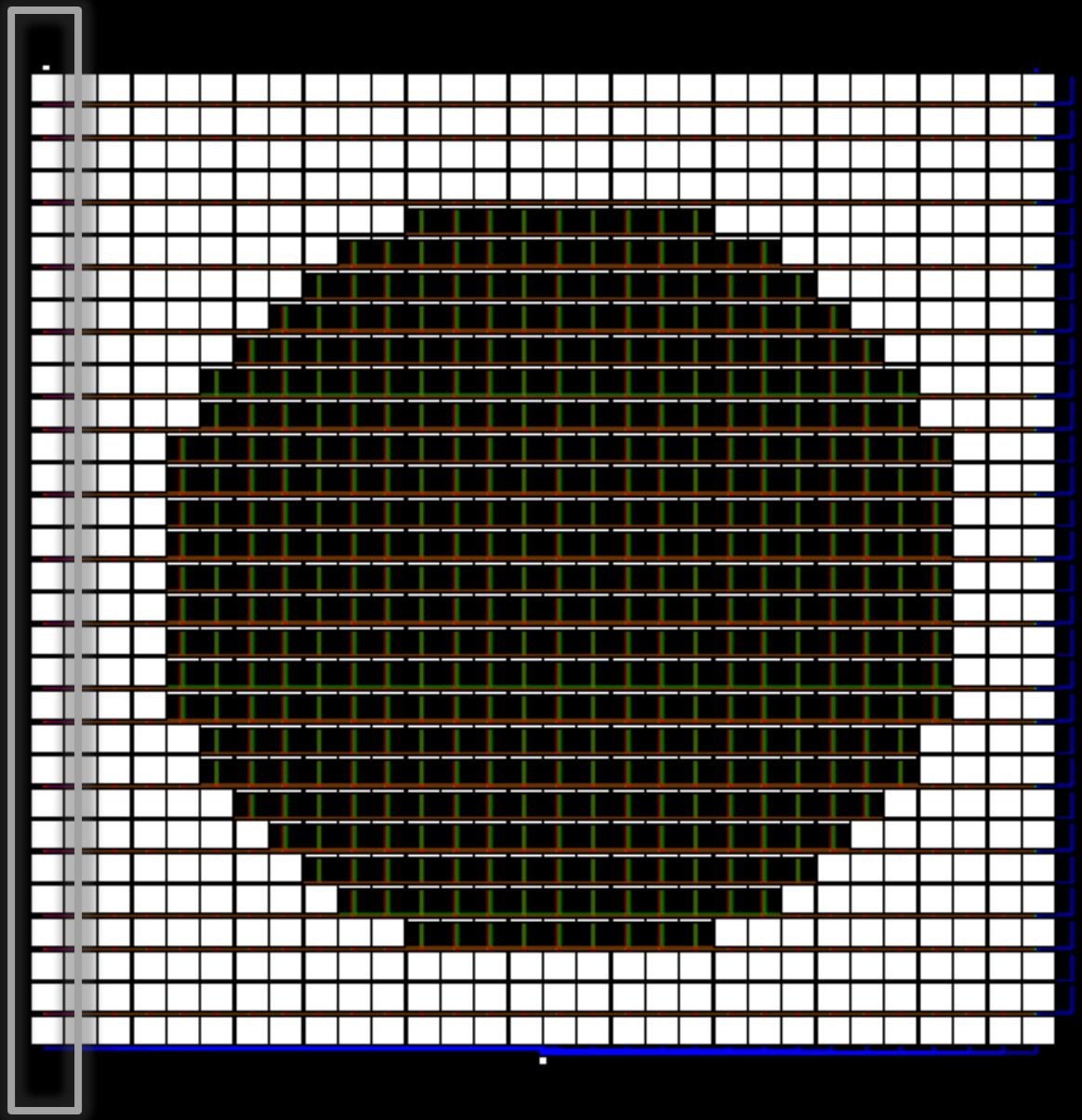

vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:(wide+4)[edi+esi*4]
jmp $+5
e_0_29:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:(wide+4)[edi+esi*4]
xor eax, eax
jz e_1_29
e_1_29:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:(wide+4)[edi+esi*4]
xor eax, eax
jz





- Insert always on column

Almost there



- Add a junk code generator

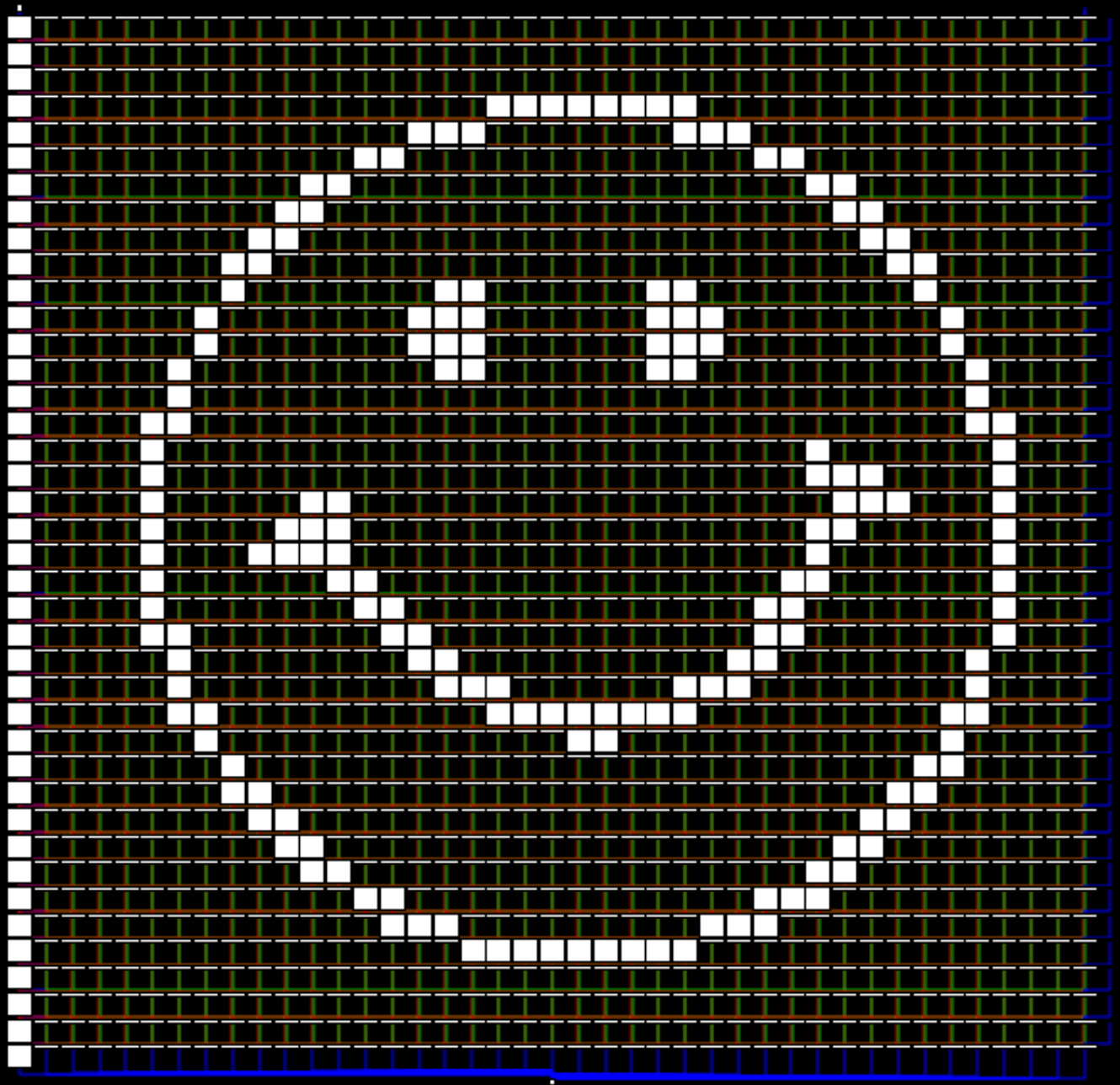
Almost there

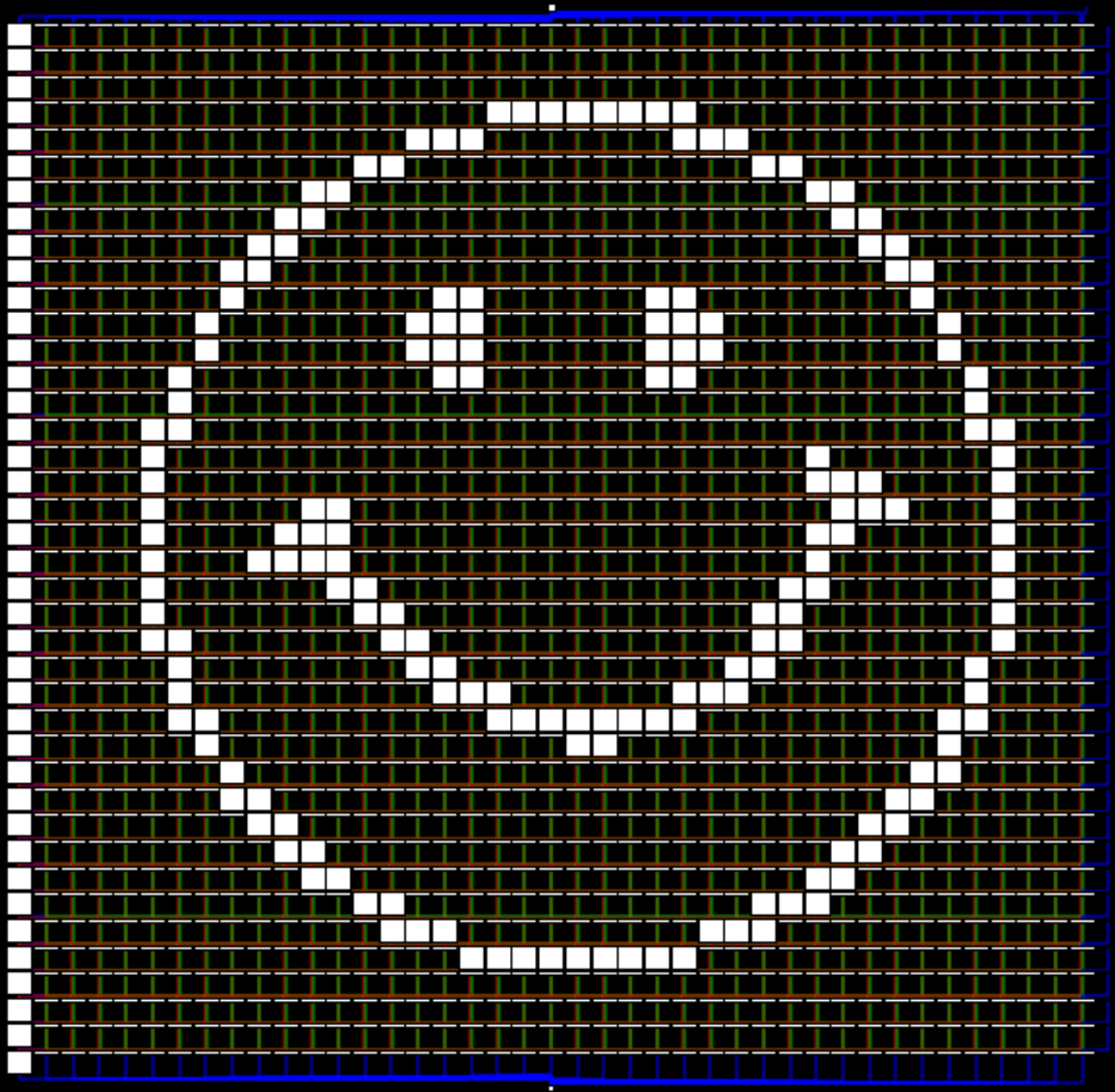
```
movzx eax, bh
movzx ecx, dh
dec ecx
xor ebx, ecx
lea ebx, [ebp+1*4]
mov eax, 3526025642
or eax, 188401817
mov ah, 4
lea eax, [ecx+4*edx]
test edx, eax
mov cl, 2
add ebx, ecx
shr eax, 21
movzx ecx, dl
add ebx, ecx
shr eax, 25
mov ah, 4
test edx, eax
shr ecx, 19
movzx eax, bh
or eax, 2742937504
mov ah, 4
and edx, eax
```

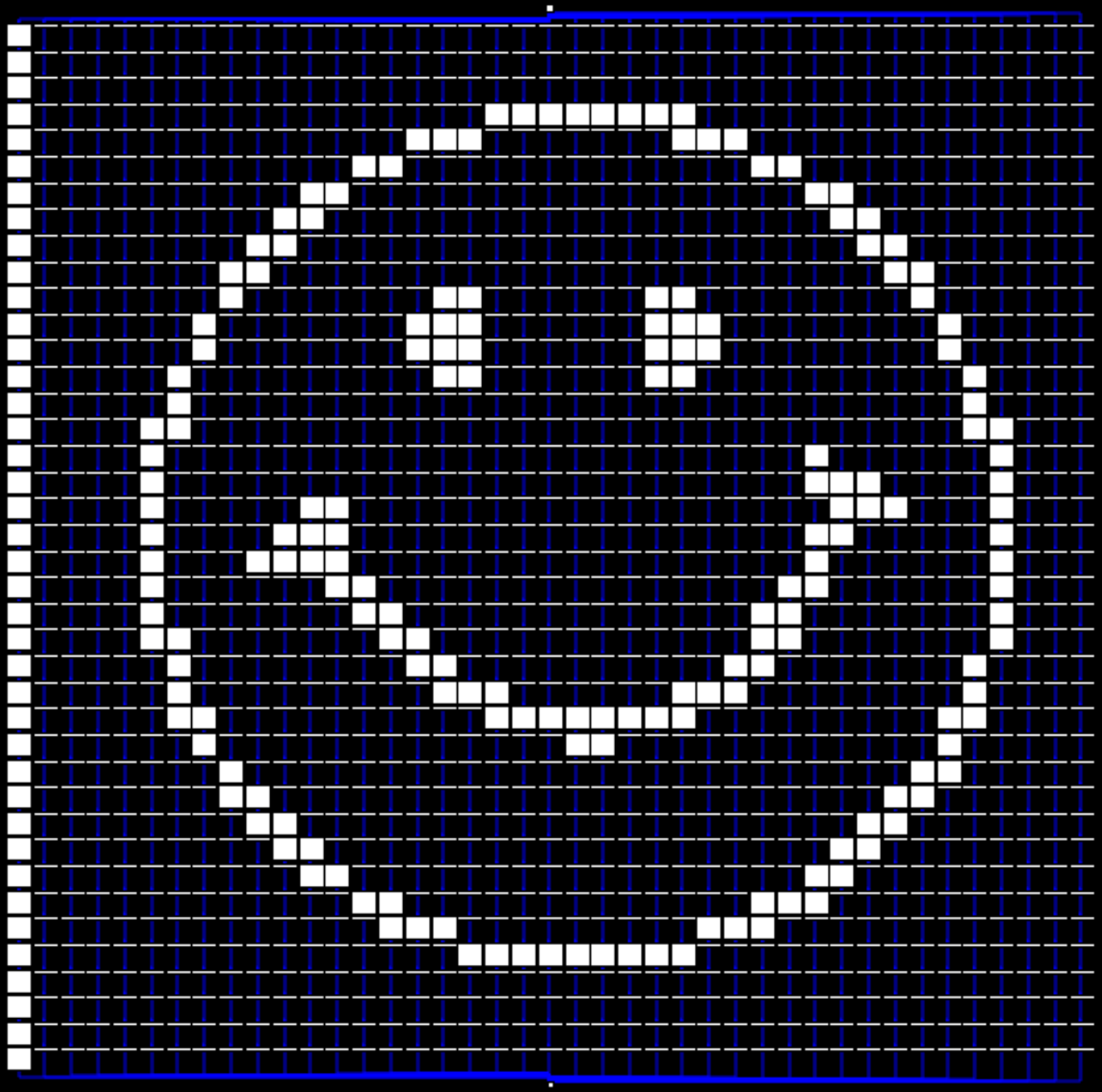
- BMP to %assign converter

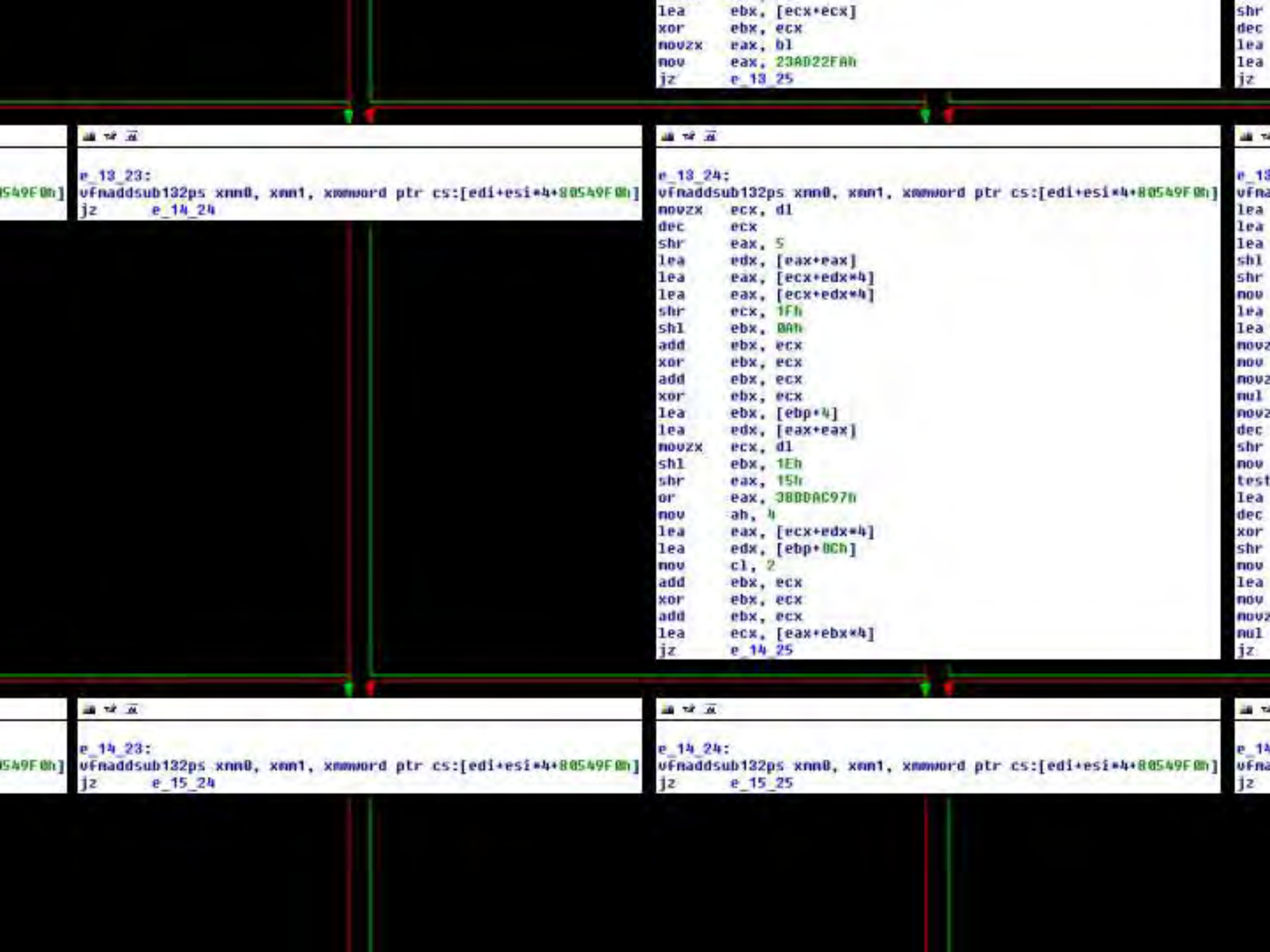
Almost there

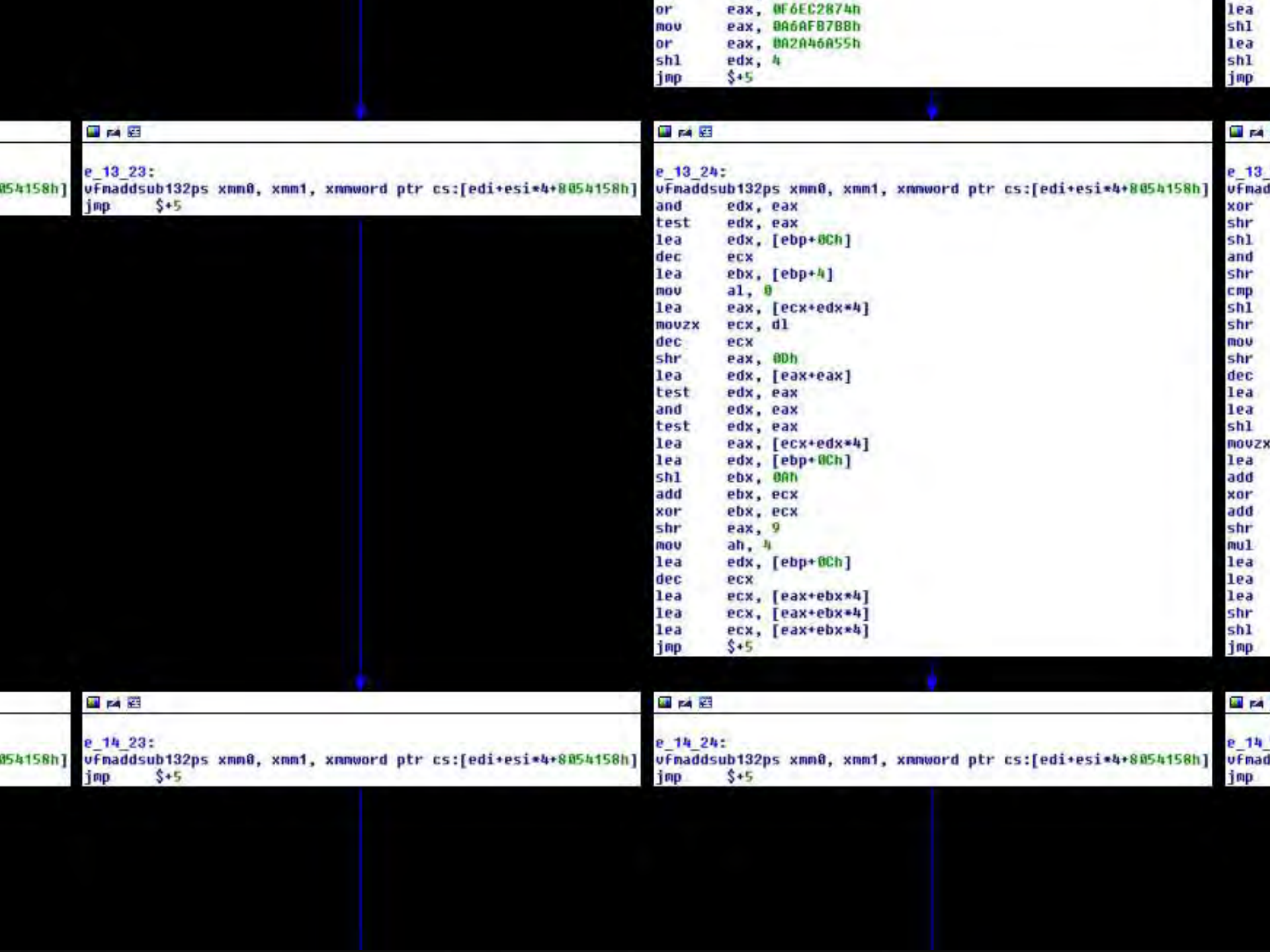
```
%assign pixel_13_5 1
%assign pixel_14_5 1
%assign pixel_15_5 0
%assign pixel_16_5 1
%assign pixel_17_5 0
%assign pixel_18_5 1
%assign pixel_19_5 1
%assign pixel_20_5 0
%assign pixel_21_5 1
%assign pixel_22_5 0
%assign pixel_23_5 0
%assign pixel_24_5 0
%assign pixel_25_5 1
%assign pixel_0_6 1
%assign pixel_1_6 1
%assign pixel_2_6 1
%assign pixel_3_6 1
%assign pixel_4_6 1
%assign pixel_5_6 1
%assign pixel_6_6 1
%assign pixel_7_6 1
```

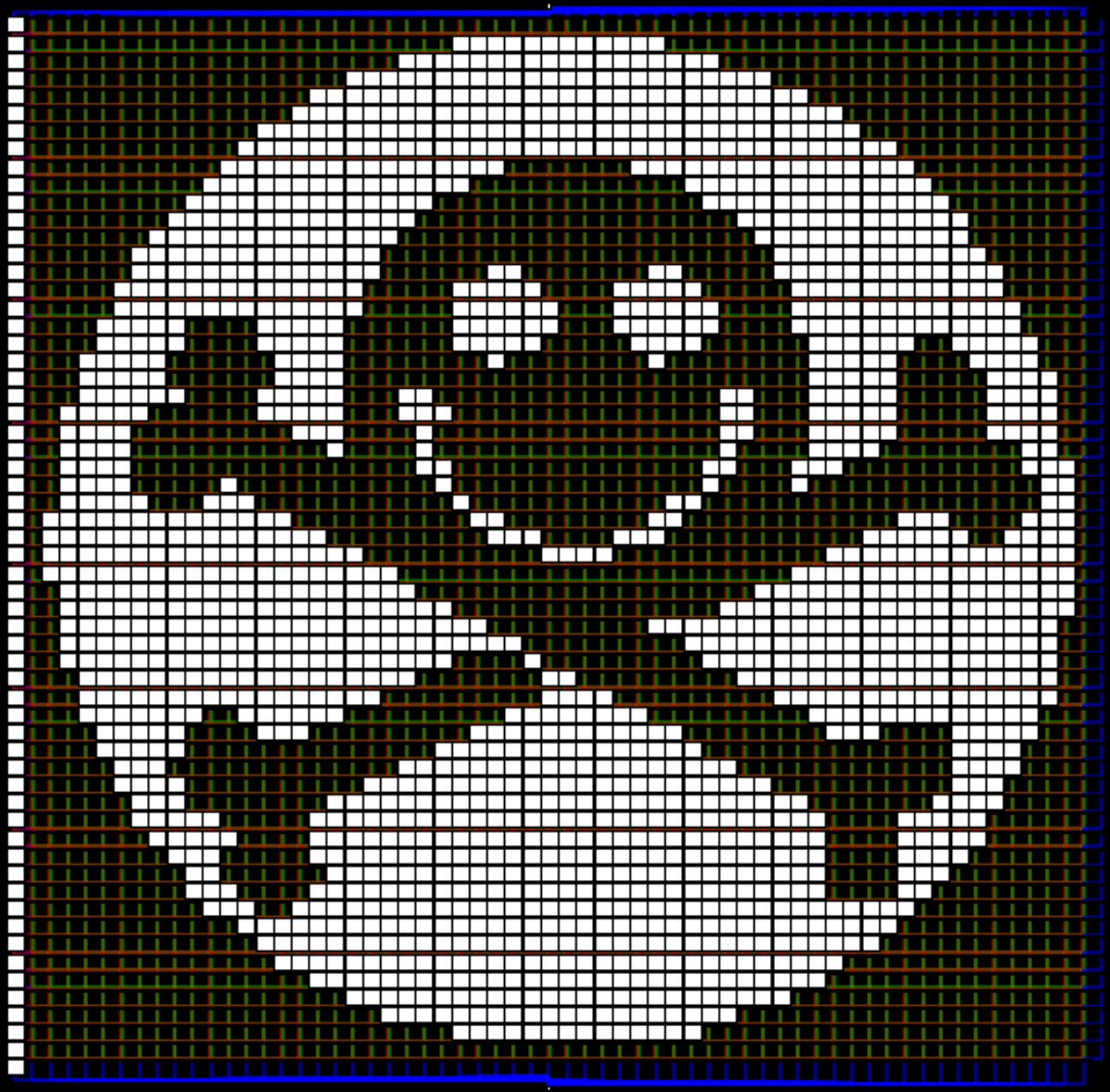






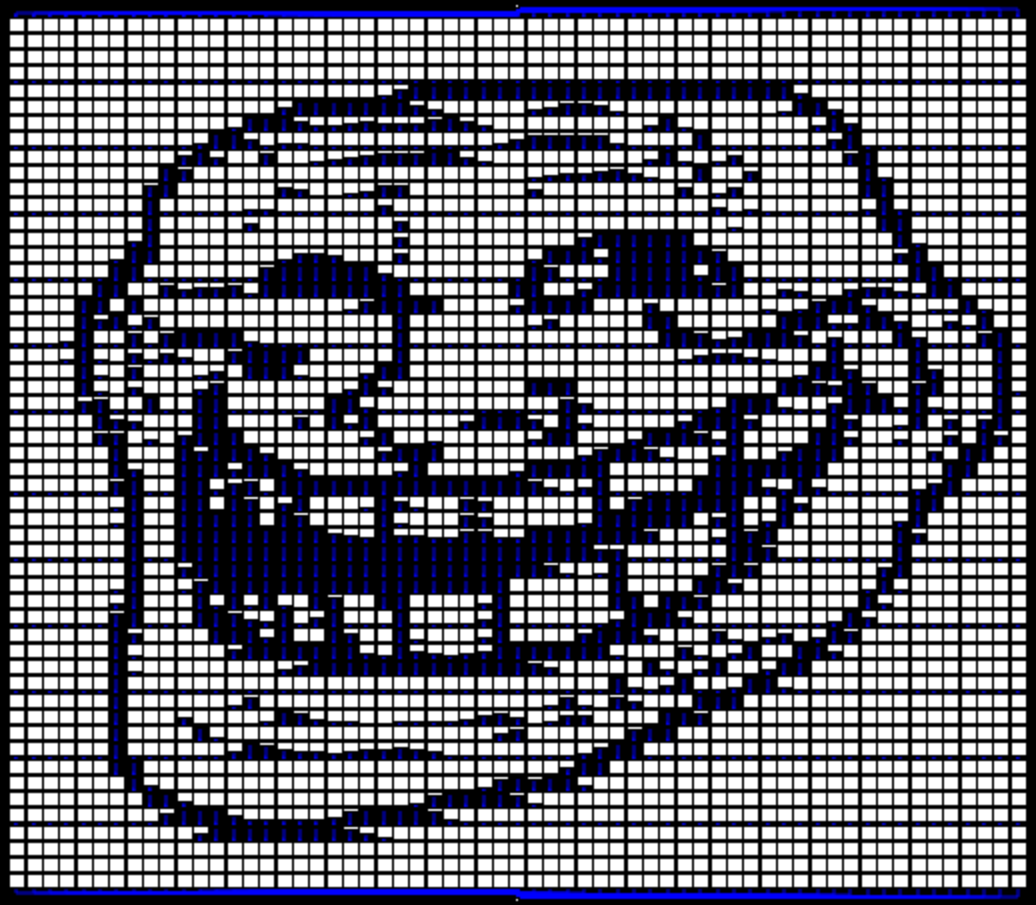

lea ebx, [ecx+ecx]
xor ebx, ecx
movzx eax, bl
mov eax, 23AD22FAh
jz e_13_25
e_13_23:
vfnaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+80549F0h]
jz e_14_24
e_13_24:
vfnaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+80549F0h]
movzx ecx, dl
dec ecx
shr eax, 5
lea edx, [eax+eax]
lea eax, [ecx+edx*4]
lea eax, [ecx+edx*4]
shr ecx, 1Fh
shl ebx, 0Ah
add ebx, ecx
xor ebx, ecx
add ebx, ecx
xor ebx, ecx
lea ebx, [ebp*4]
lea edx, [eax+eax]
movzx ecx, dl
shl ebx, 1Eh
shr eax, 15h
or eax, 38DDAC97h
mov ah, 4
lea eax, [ecx+edx*4]
lea edx, [ebp+0Ch]
mov cl, 2
add ebx, ecx
xor ebx, ecx
add ebx, ecx
lea ecx, [eax+ebx*4]
jz e_14_25
e_14_23:
vfnaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+80549F0h]
jz e_15_24
e_14_24:
vfnaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+80549F0h]
jz e_15_25

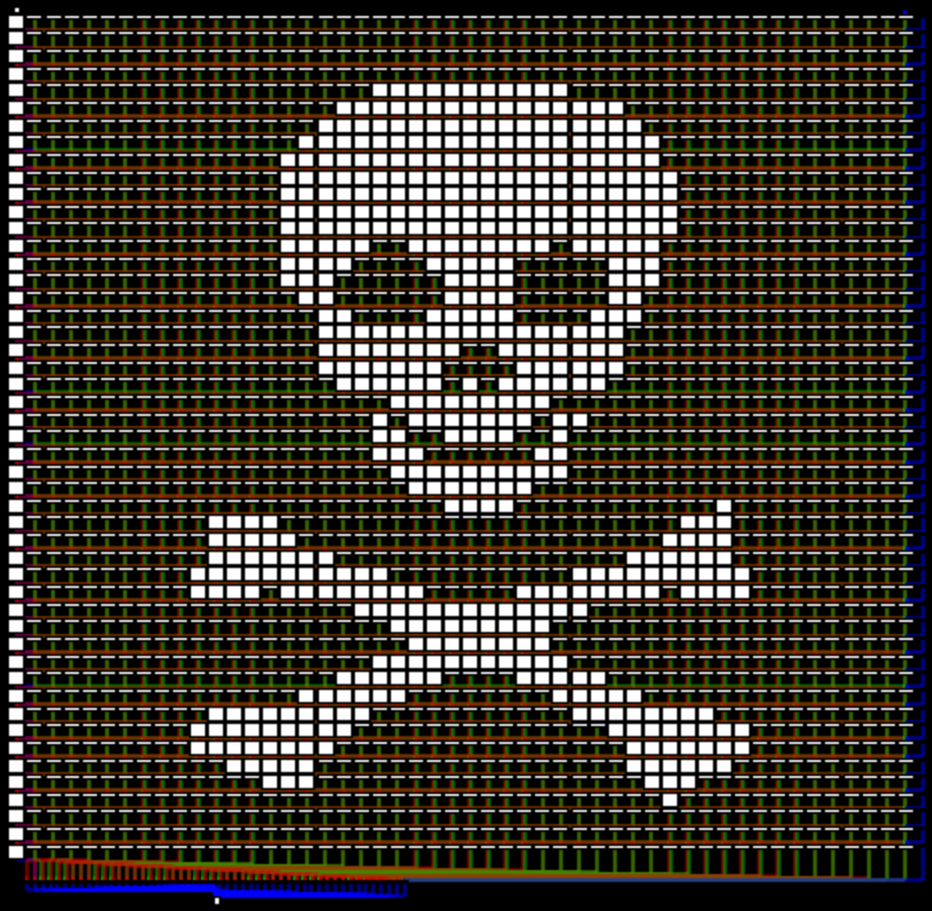

or eax, 0F6EC2874h
mov eax, 0A6AFB788h
or eax, 0A2A46A55h
shl edx, 4
jmp $+5
e_13_23:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8054158h]
jmp $+5
e_13_24:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8054158h]
and edx, eax
test edx, eax
lea edx, [ebp+0Ch]
dec ecx
lea ebx, [ebp+4]
mov al, 0
lea eax, [ecx+edx*4]
movzx ecx, dl
dec ecx
shr eax, 0Dh
lea edx, [eax+eax]
test edx, eax
and edx, eax
test edx, eax
lea eax, [ecx+edx*4]
lea edx, [ebp+0Ch]
shl ebx, 0Ah
add ebx, ecx
xor ebx, ecx
shr eax, 9
mov ah, 4
lea edx, [ebp+0Ch]
dec ecx
lea ecx, [eax+ebx*4]
lea ecx, [eax+ebx*4]
lea ecx, [eax+ebx*4]
jmp $+5
e_14_23:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8054158h]
jmp $+5
e_14_24:
vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8054158h]
jmp $+5



- REpsych Toolchain
  - Generates assembly …
    … to form images through CFGs
  - (Demo)

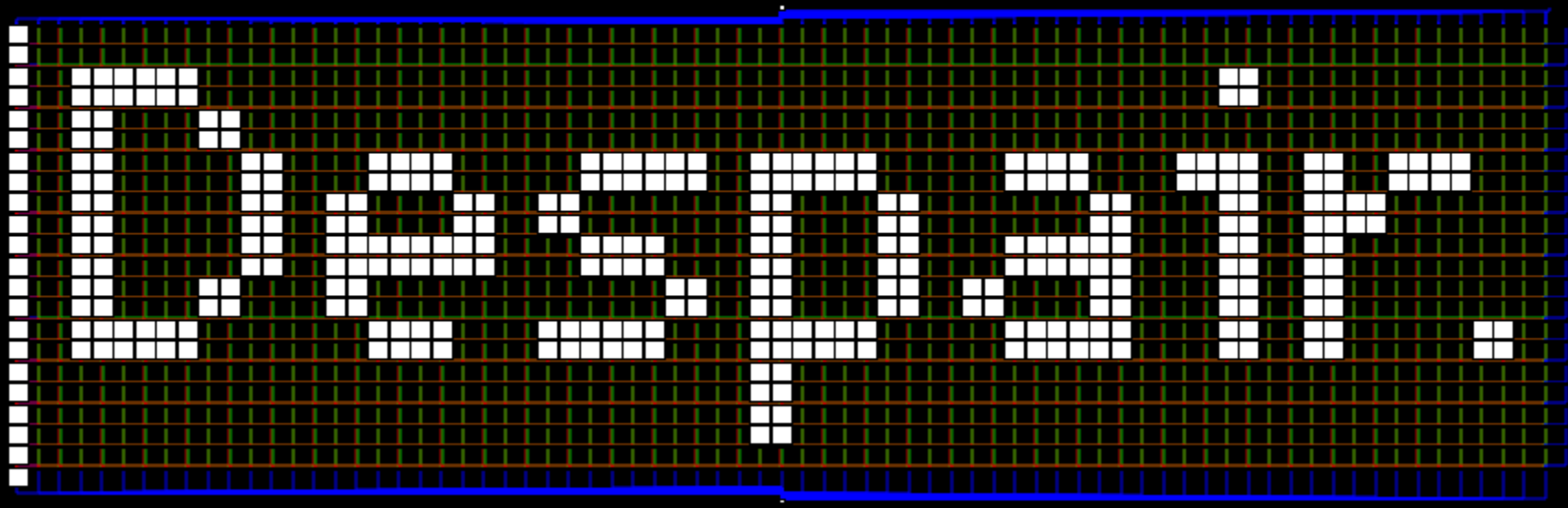
REpsych

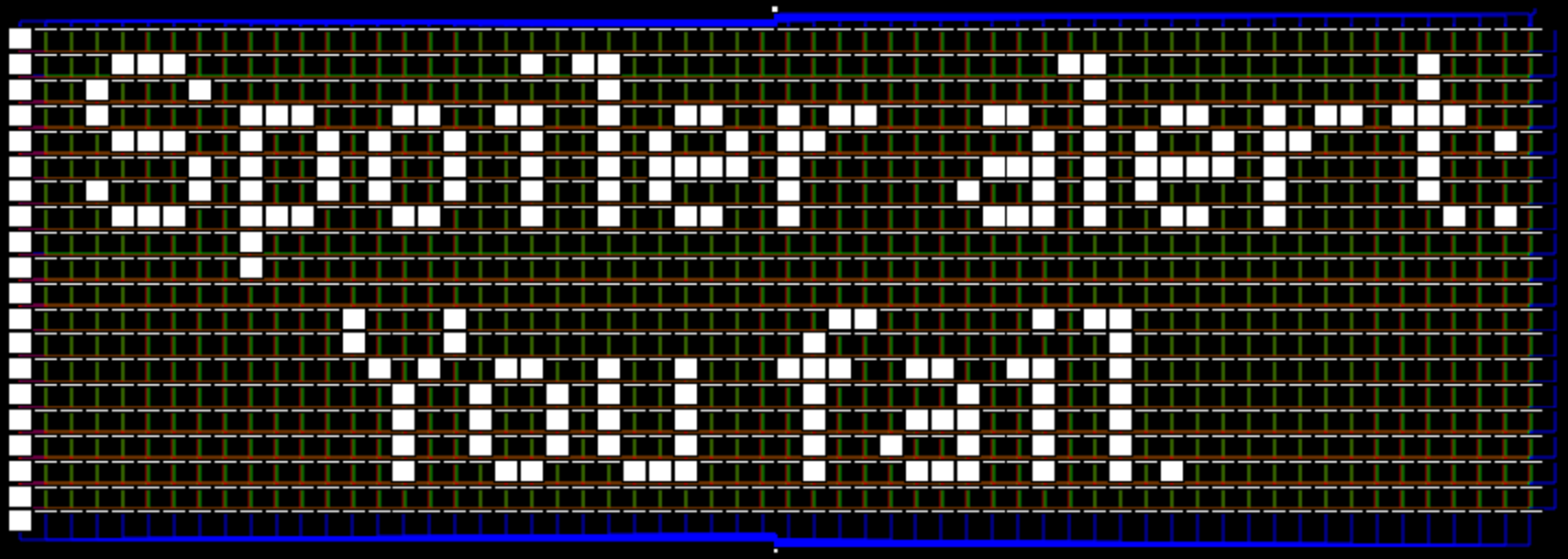




- Reverser is forced to sit and stare at whatever message you embed
- Use it to your advantage, crush their soul

Psychological Warfare

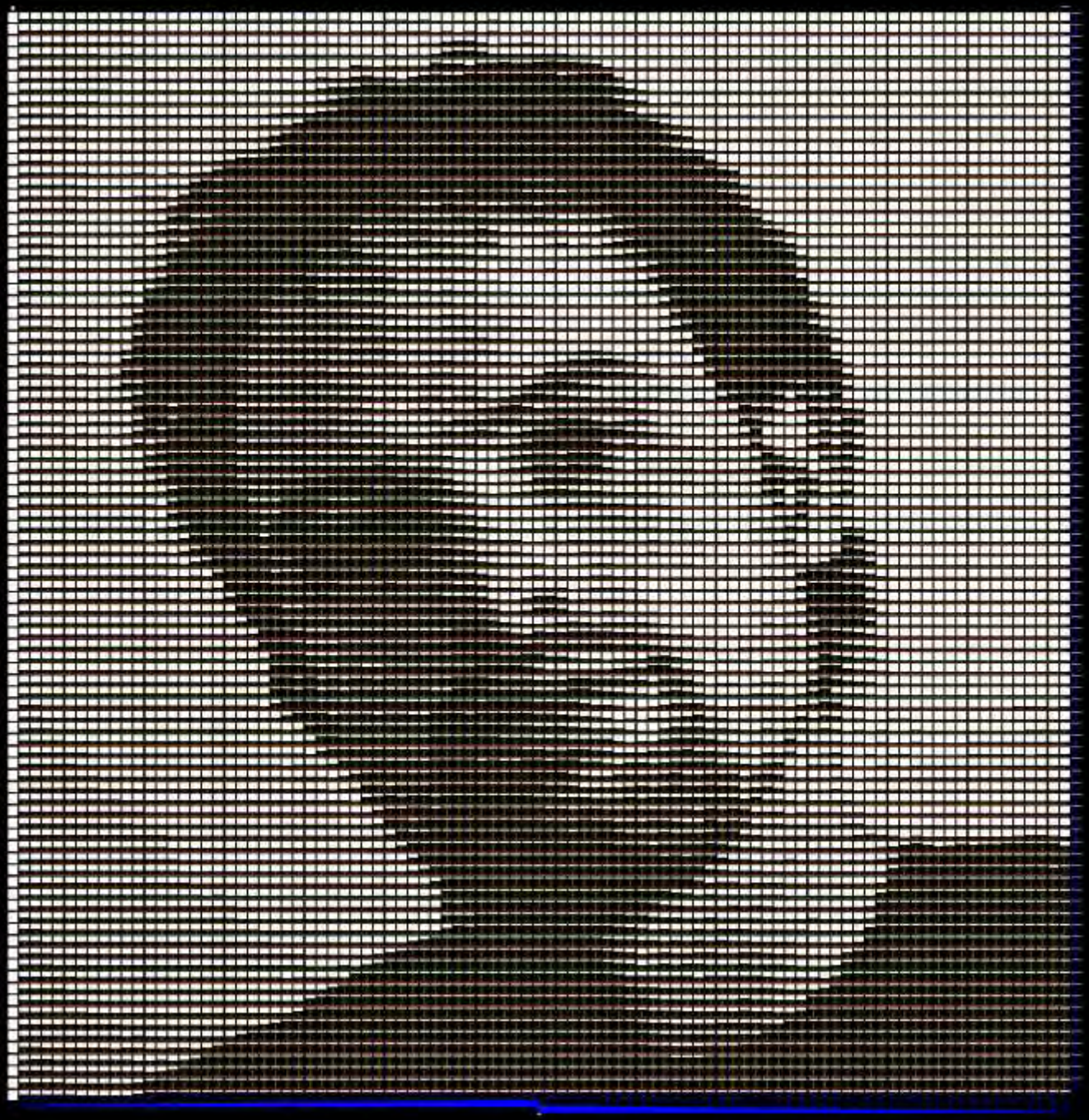

Loading...



The gods worse and worry,
until you give up or die.

- (Draw an assembly selfie)

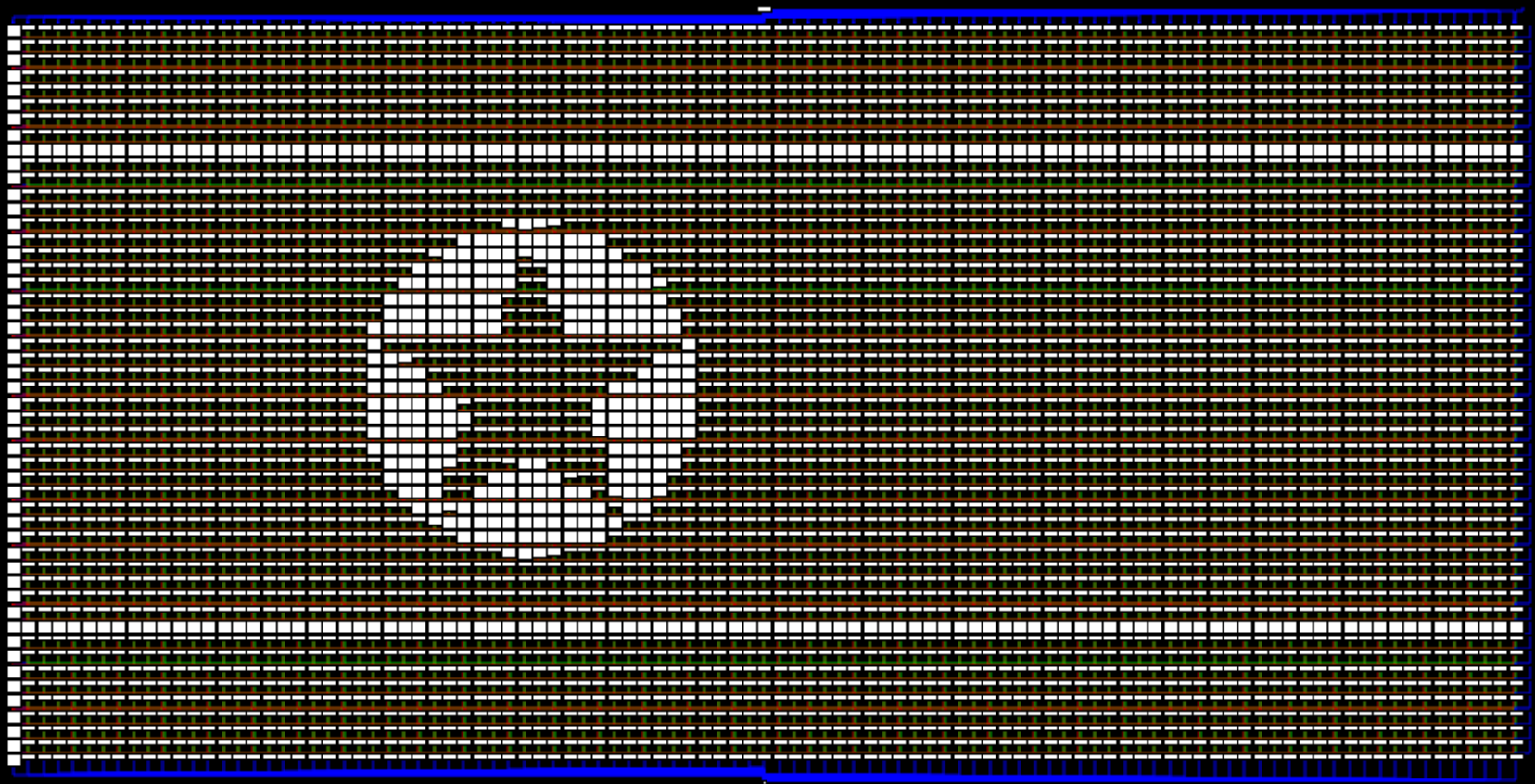
Grayscale



Stego

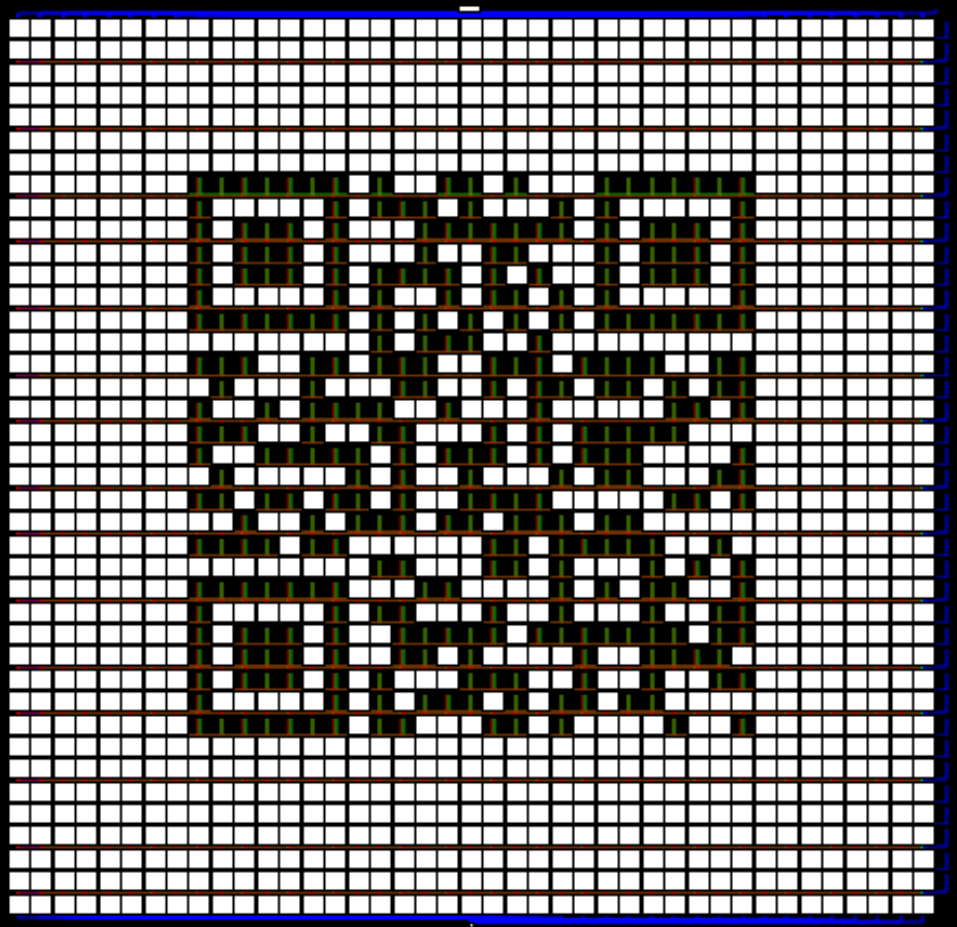
the_interview.exe

More ideas



More ideas

- QR
  - a.k.a. the ultimate CTF problem



- Creepiest malware ever
  - Scans your hard disk
  - Rewrites itself to match your personal images
  - (Demo)

More ideas

- 14 lines of assembly
- 328 lines of preprocessor macros

- github.com/xoreaxeaxeax
  - **REpysch**
  - M/o/Vfuscator 2.0
  - x86 0-day POC
  - Etc.

- Feedback?

- domas
  - @xoreaxeaxeax
  - xoreaxeaxeax@gmail.com