SMW Jailbreak Injection Route

By SethBling and Cooper H.

This document contains notes for the execution of a code injection exploit which will permanently install a hex editor into a Super Mario World cartridge's save files.

Setup

You will need:

- A NTSC SNES/Super Famicom
- Two USA copies of Super Mario World (second copy optional)
- Two multitaps
- Three controllers
- Some way to hold down buttons on two of the controllers

Plug a multitap into each of the console's two controller ports, and make sure they're both in 3-5P mode.

- For the multitap in the console's first controller port:
    - Plug your main gameplay controller into the first multitap port.
    - Plug a controller into the second multitap port with these buttons held down:
        - L, B, Y, Select, Down (`20 E4`)
- For the multitap in the console's second controller port:
    - Plug a controller into the second multitap port with these buttons held down:
        - Select, Y (`00 60`)

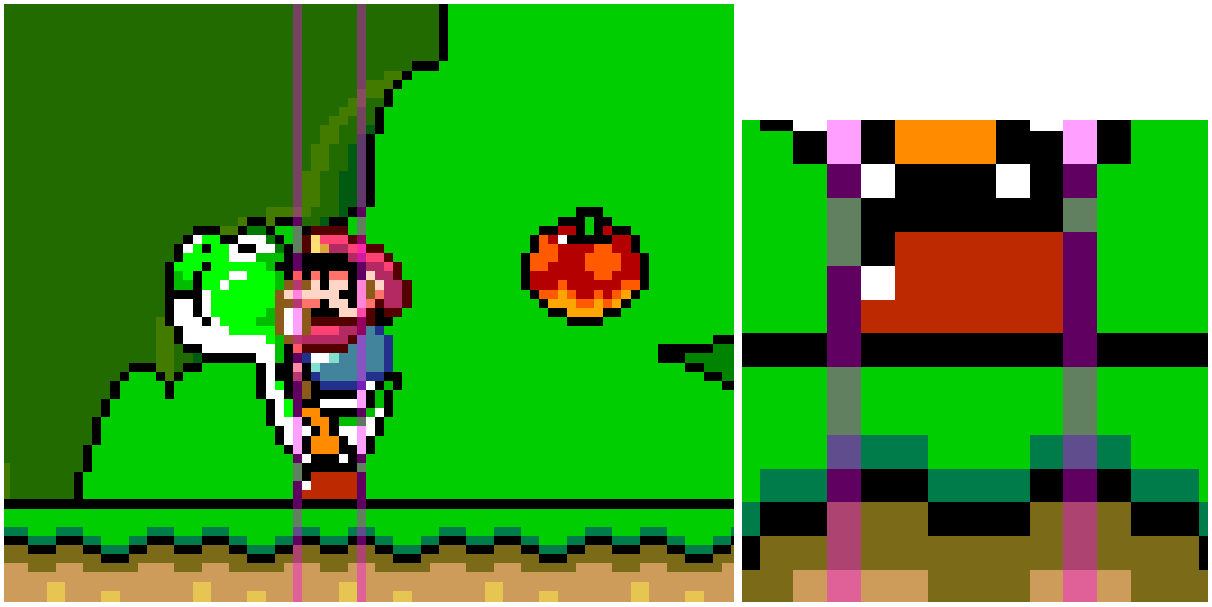
Route Overview

1. Use Powerup Incrementation (PI) to get Mario's powerup state to 6.
2. Use Arbitrary Code Execution (ACE) to change Mario's powerup state to 22 (PI22).
3. Use ACE to extend the timer.
4. Set up multi-byte write ACE with the P Switch.
5. Iteratively write the primitive hex editor (bootstrap) to OAM Subroutine with PI22 ACE, using the P Switch as the data value.
    a. First write the coin display.
    b. Then write the rest of the bootstrap.
6. Use the hex editor to write the jailbreak to unused stack space.
    a. The jailbreak installer will automatically run when the last byte is written.
7. Hard power cycle the console and load the corrupted save file.
8. Use the hex editor to write a basic payload demonstrating the exploit has worked.
9. Hard power cycle the console and load the corrupted save file.

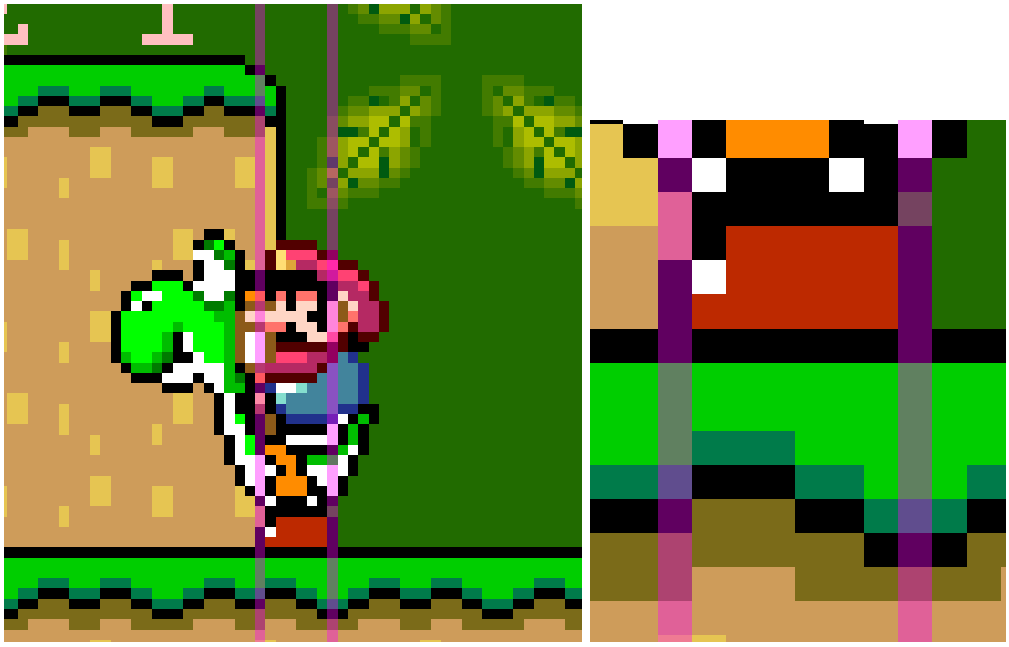
10. Use the hex editor to write a payload to copy the jailbreak into unused stack space.
11. Without powering off the console, switch carts to the fresh SMW cartridge and soft reset.
12. Use RLX ACE to run the save file installer on the second cartridge.
13. Hard power cycle the console and load the corrupted save file.
14. Use the hex editor to write another basic payload demonstrating the exploit has worked.

Powerup 0x16

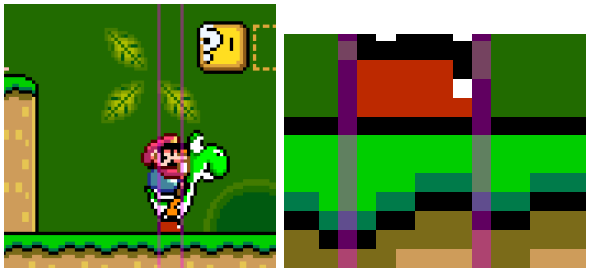
84 STY (screen 0)



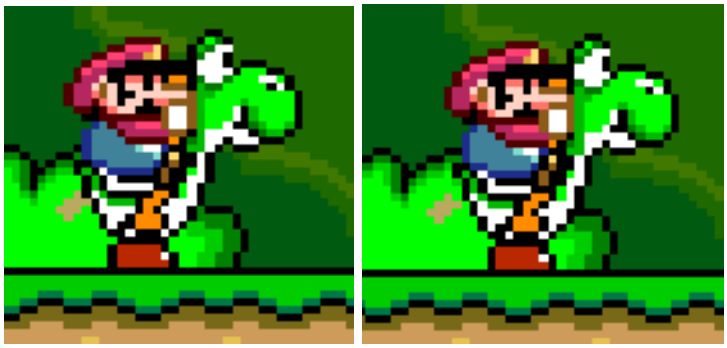
19



60 RTS

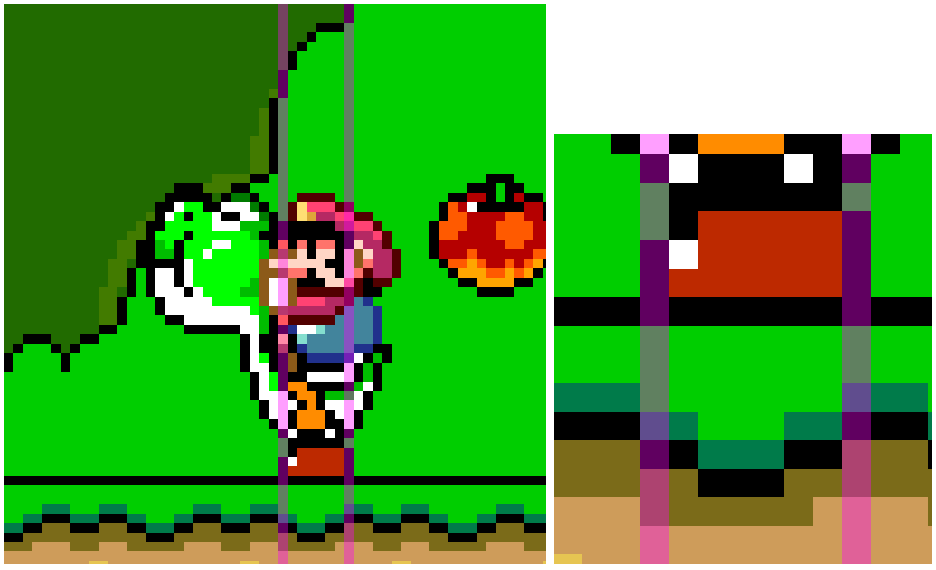


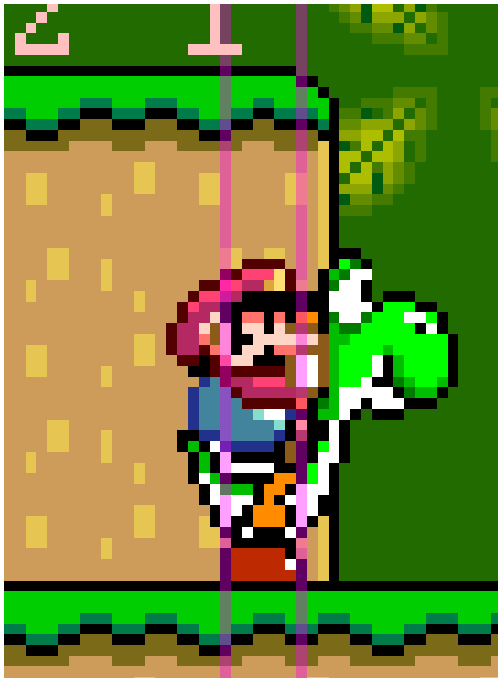
Scroll the screen left and hit the 1Up block. Walk right until standing on one of these two spots (don't overshoot, the screen scroll is what's important)



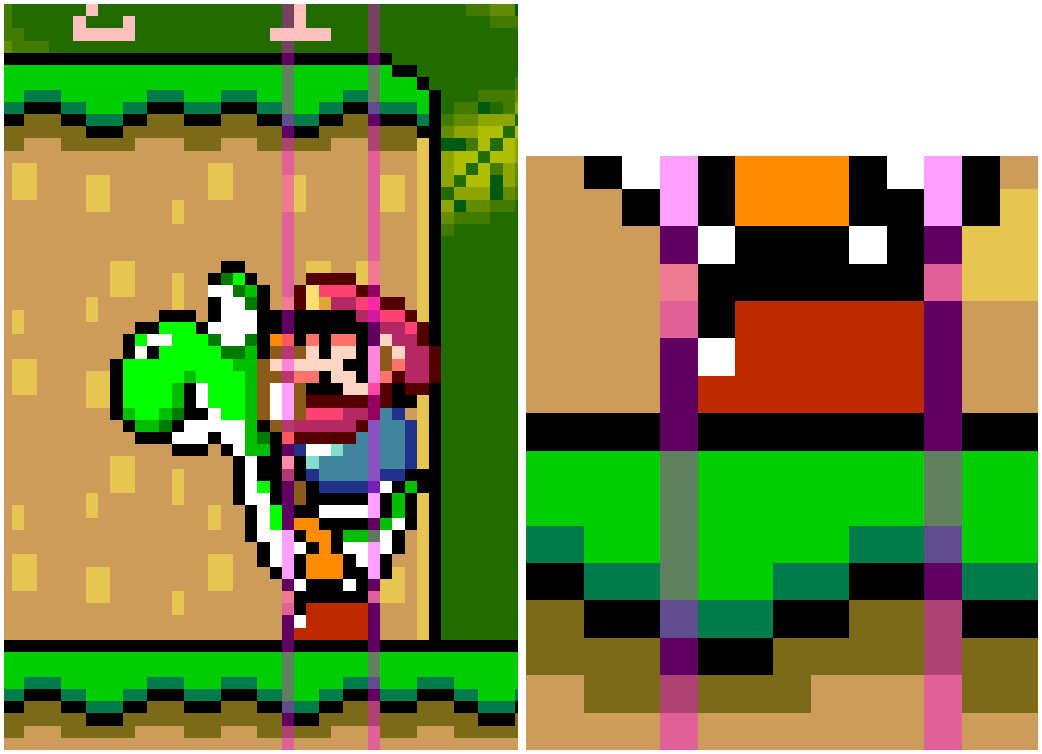
PI22 ACE to Extend Timer

Get 11,900 in-game seconds on the clock:
8D STA (this is on screen 0)

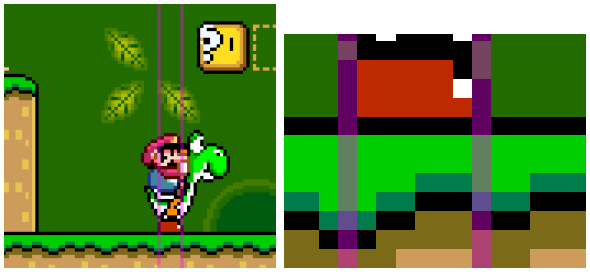




0F



60 RTS



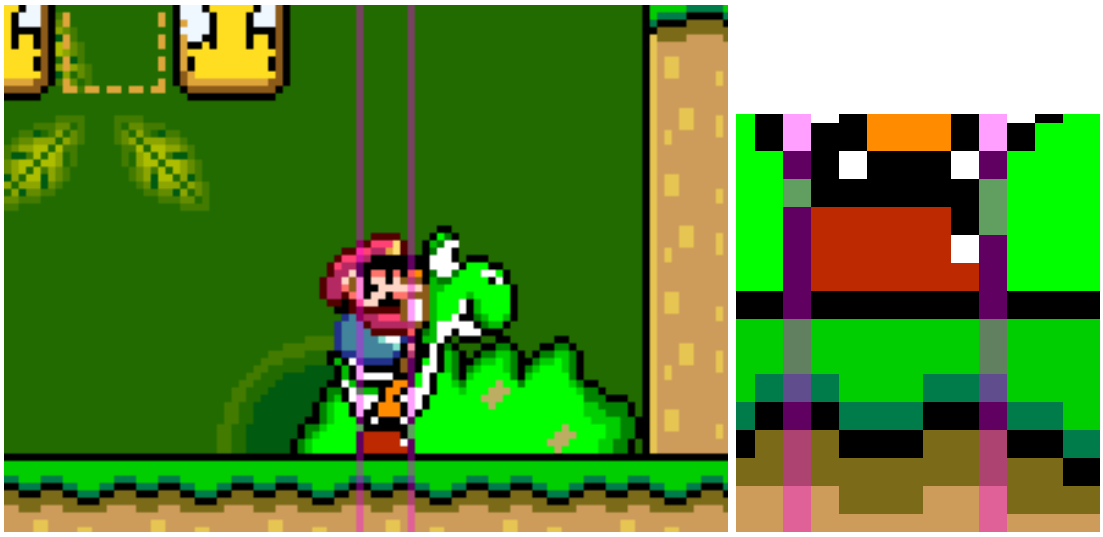
Make sure there are **two glitched berries**!
Get the 1Up from the Yoshi block in screen 7 while holding A.
Get the 1Up on the first ledge in screen 7 while holding A.

Multi-Byte Write ACE Setup

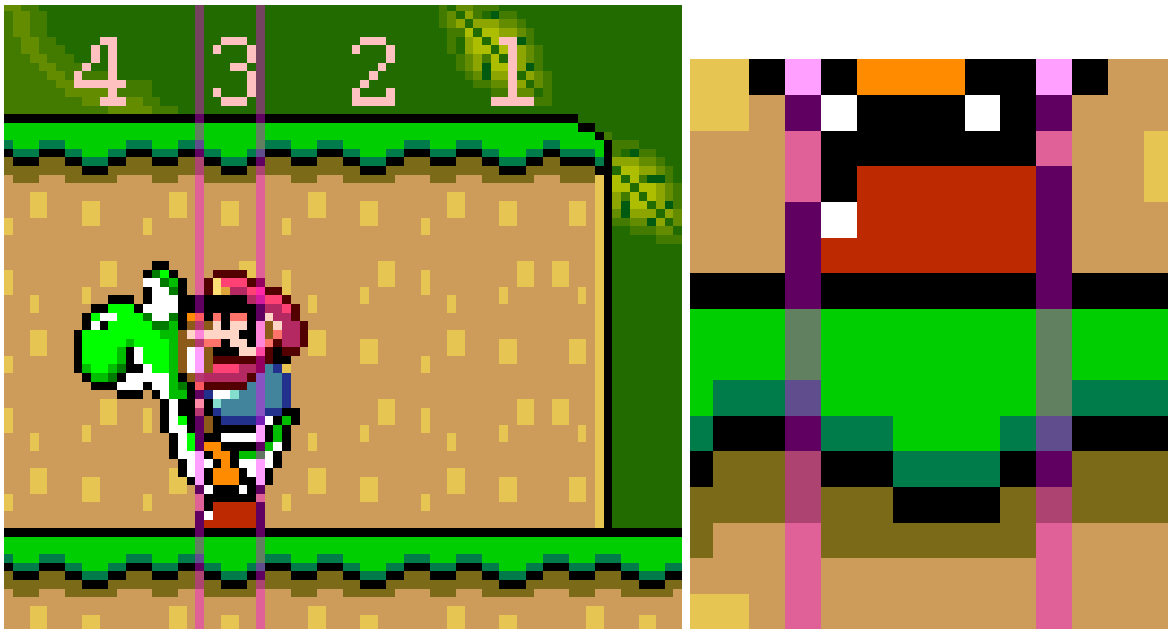
- Jump off Yoshi
- Go to sublevel
- Come back
- Grab P Switch (slot 8)
- Get Yoshi from rightmost Yoshi block (slot 9)
- Glitch 4 berries (Slots B, A and 7, 6)

        - Make sure at most one enemy is on screen for berry slot 7 and no enemies are on screen for berry slot and 6
- Take a hit from a koopa so Yoshi runs off screen
- Destroy the shell on the ground in screen 0
- Grab Yoshi from block (Slot 5 because message box is slot 9)
- Eat the two most recently glitched berries (Slots 7 and 6)
- Program slots 0-7:
    - LDA $EC; STA [_$E9_]; RTS; $7F81XX; YY
        - A5 (LDA)
        - EC
        - 87 (STA)
        - E9
        - 60 (RTS)
        - XX (Yoshi)
        - 81
        - 7F
        - YY (P Switch)
- Place Yoshi for address low byte
- Place P Switch for value
- ACE
- Repeat

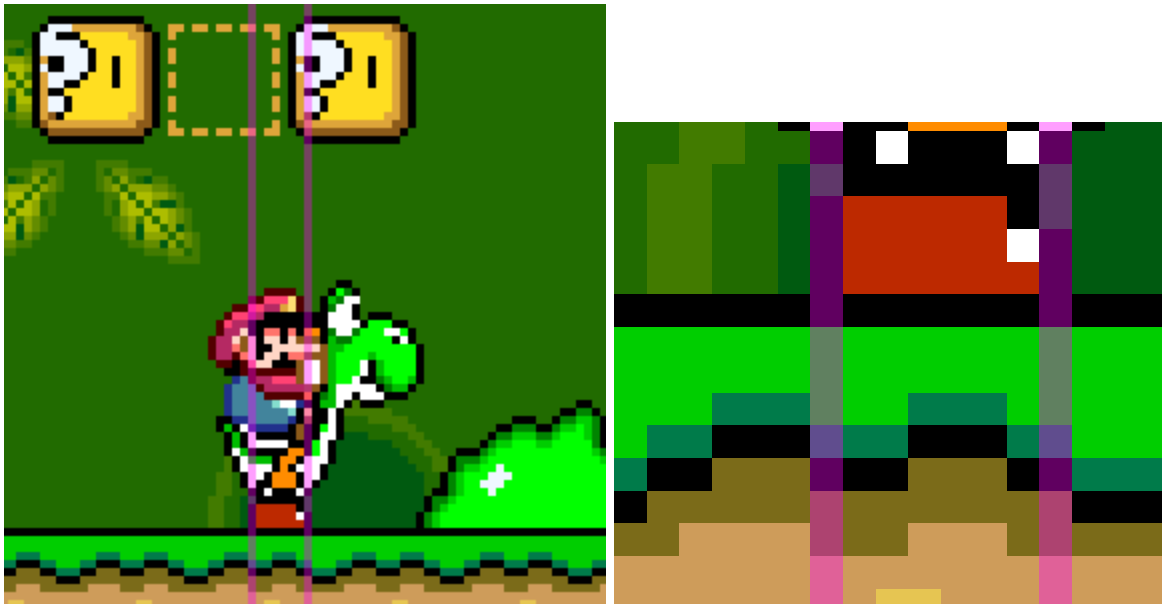
A5 (LDA)



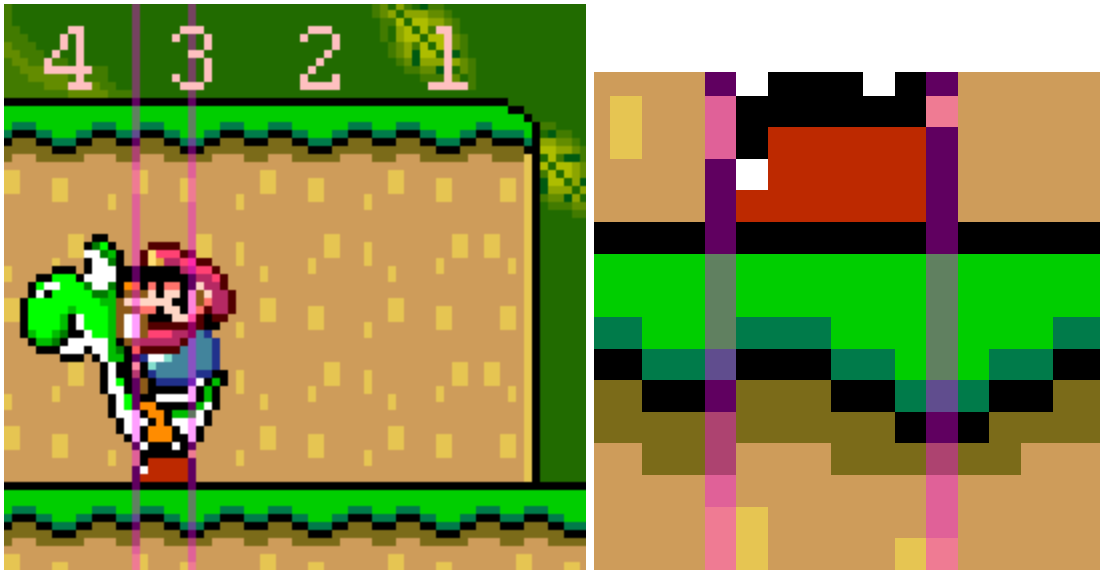
EC



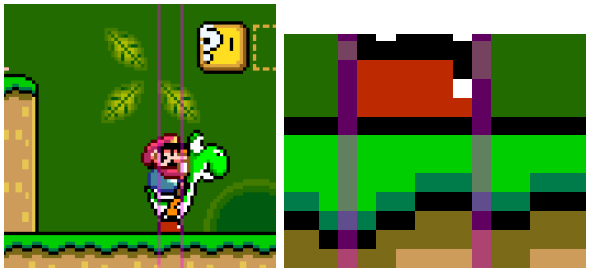
87 (STA)



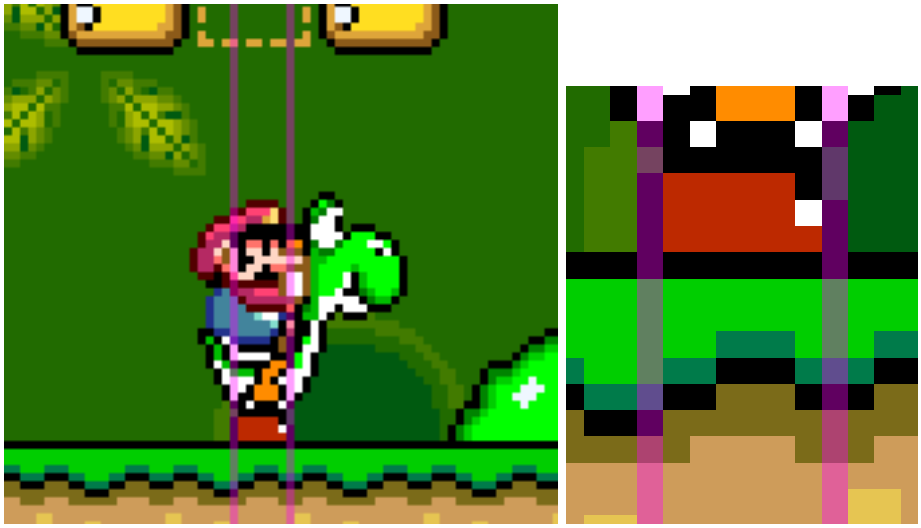
E9



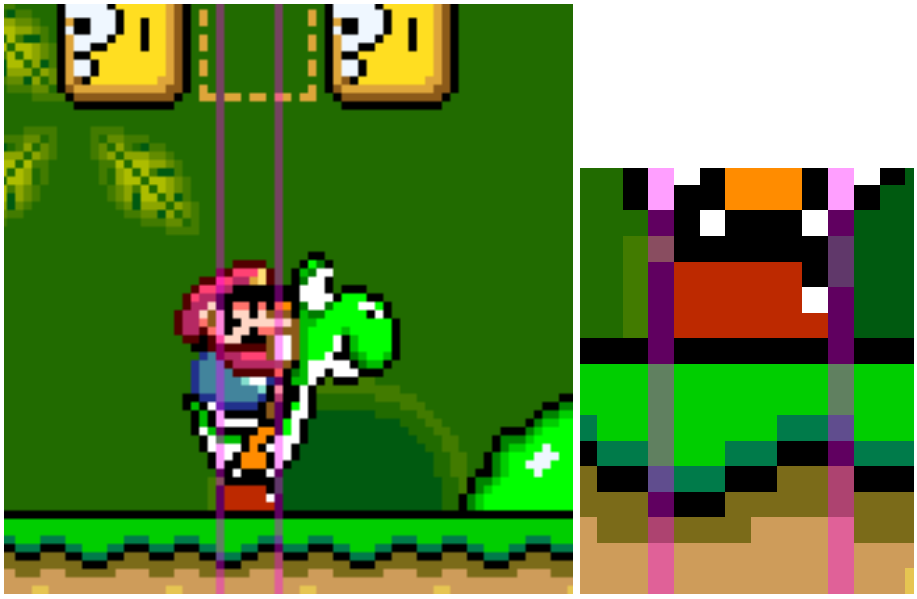
60 (RTS)



81



7F

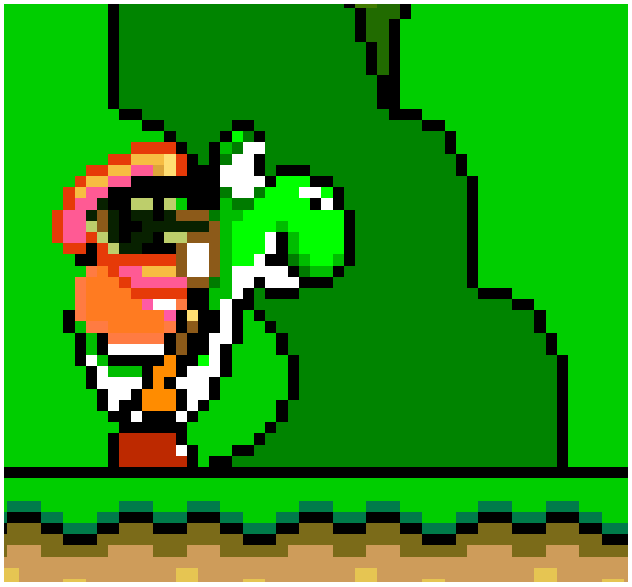


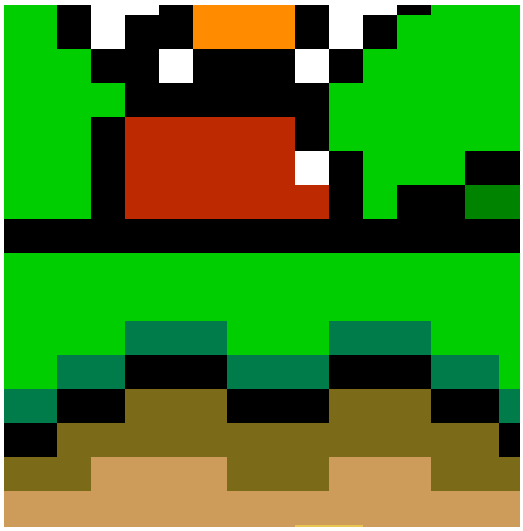
There should now be 1 red koopa left.

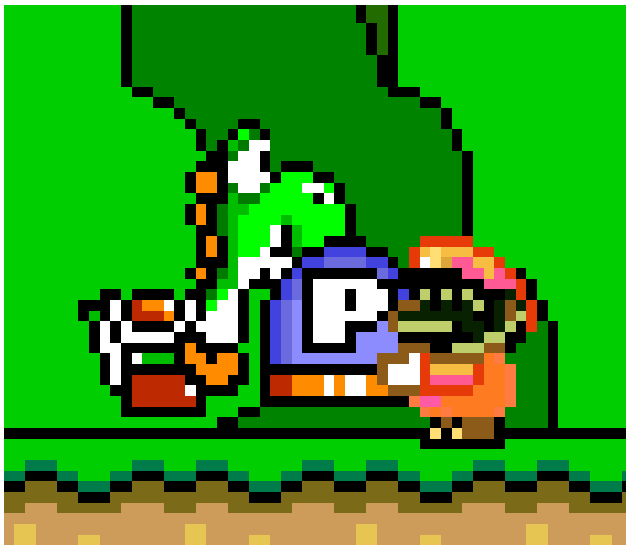
Multi-Byte Write

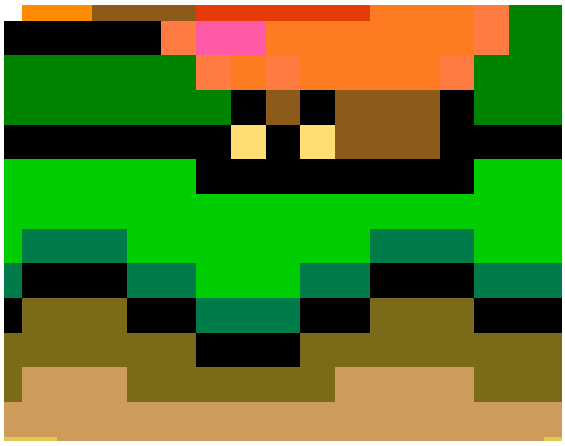
Bootstrap Coin Display

Code at 7F8182: A5 94 8D BF 0D 6B 83 94 (Overlap)

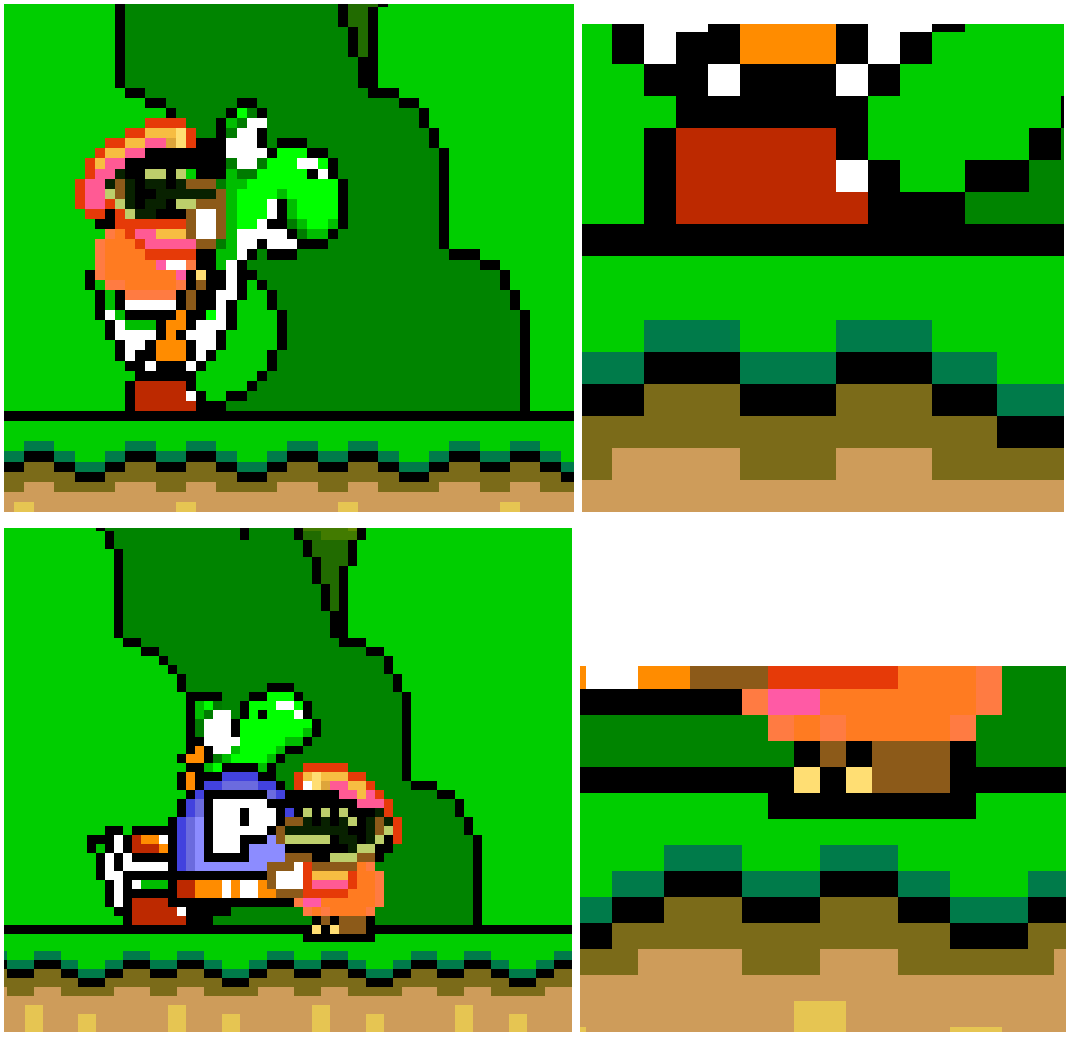




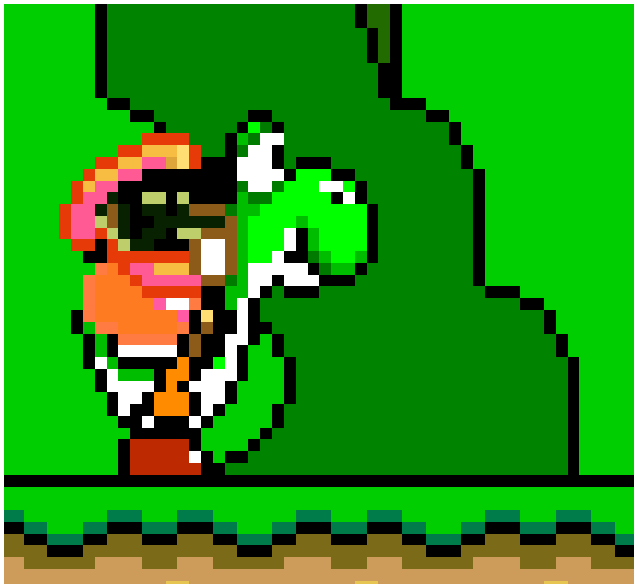


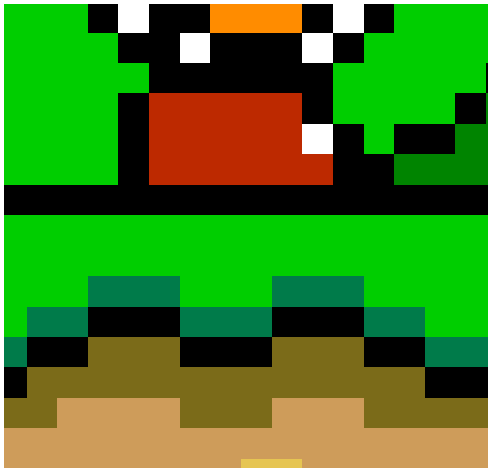


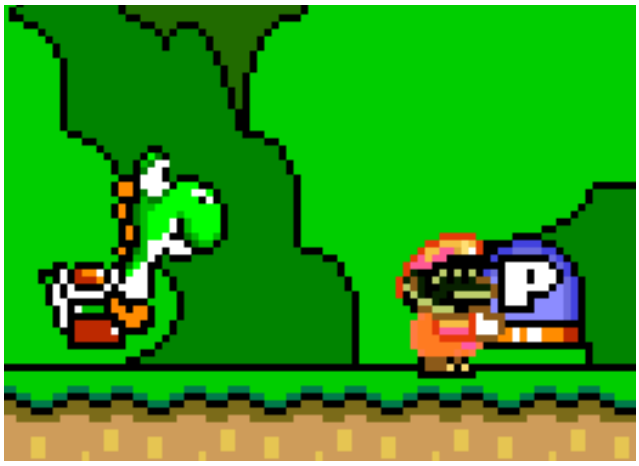
84 8D (Overlap)



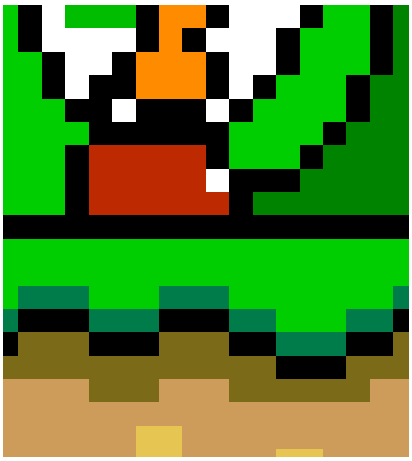
85 BF

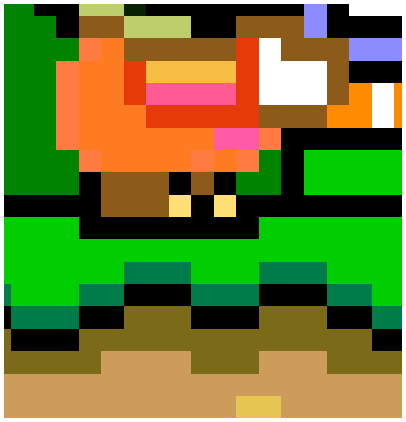




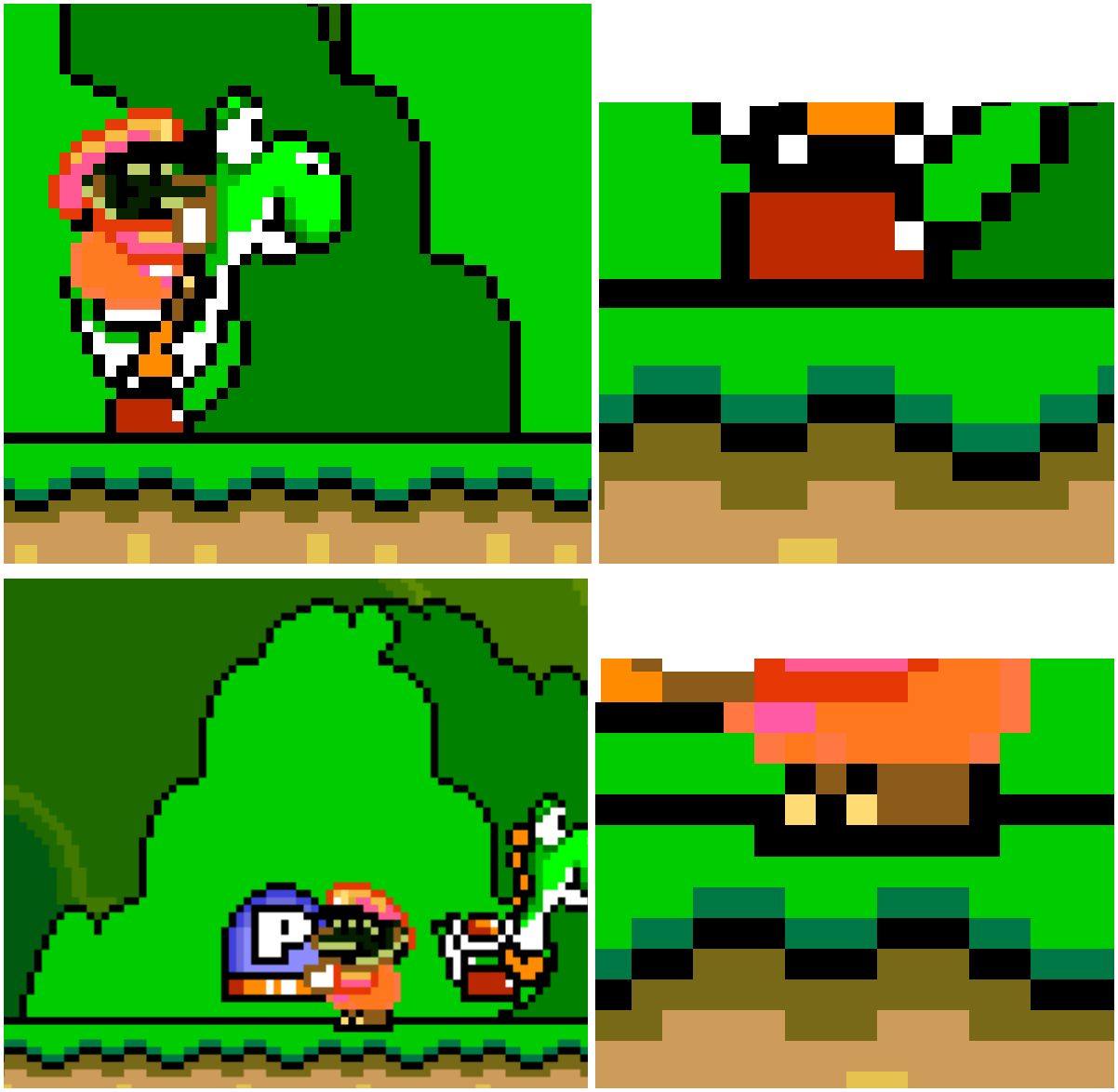


86 0D

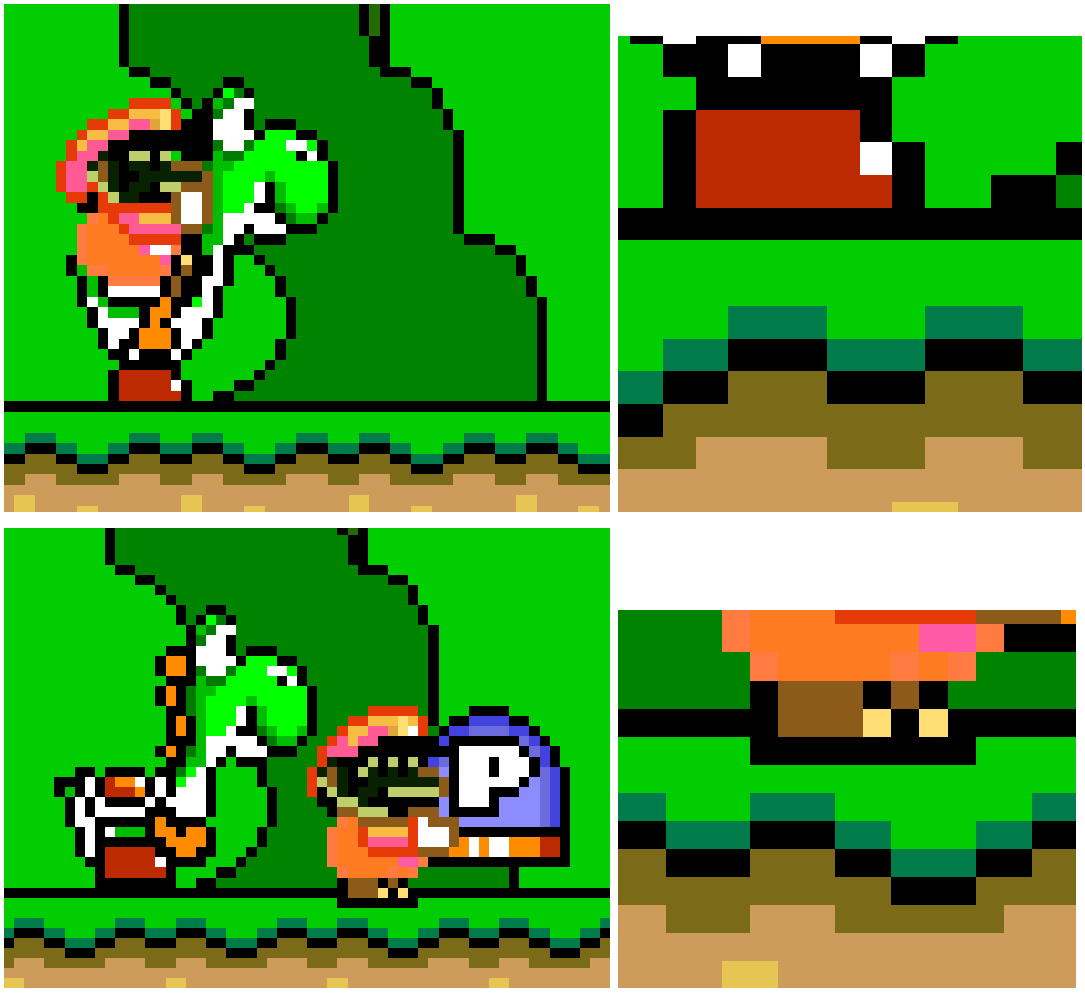




87 6B



82 A5 (Final Byte to Write)



Rest of the Bootstrap

Code at $7F8187: **A6 18 D6 6C A6 EC F0 76 8E 48 0F 9F FF 81 7F 6B**

```
LDA $94; STA $0DBF; LDX $18; INC $6C,X; LDX $EC; BEQ $7E; STX $0F34;
STA $7F8201,X; RTL
```

```
D8 M5
D9 M9
D5 N5
E5 O6
E3  4
D4 13
E2 61

E8 96
```

```
D6 97
E0 B5
E7 B6
E6 B9
E1 D1
E4 E8
D7 F5

D3 F5 (Final byte, triggers payload)
```

Initial Jailbreak Creation

Move the P switch to index 12 or less without going past 0 (Coin Display should be around 23).

```
12: O4
13: B2
14: G3
15: H1
16: C2
17: E0
18: E1
19:  1
20: J4
21: 32
22: G9
23: I3
24: 89
25: E1
26: H1
27:  1
28: G9
29: N1
30: P5
31: D6
32: F3
33: H5
34:  3
35: D6
36: F3
37: H5
38:  3
39: D6
```

40: K8
41: O5
42: G2
43: 3
44: J1
45: 65
46: D0
47: C7
48: F7
49: 79
50: 4
51: K2
52: 16
53: O6
54: J1
55: 0
56: D0
57: C7
58: F7
59: 0
60: P4
61: K2
62: C8
63: O6
64: 3
65: 92
66: 90
67: P4
68: P4
69: 41
70: 15
71: 0
72: 9
73: 0
74: 56
75: J6
76: P1
77: K8
78: 3
79: 9
80: 0
81: 4
82: K2

83: F9
84: C9
85: D1
86: C7
87: K2
88: 96
89: G4
90: O9
91: K8
92: 7
93: N0
94: O9
95: G9
96: 3
97: E1
98: 0
99: 1

100: G9
101: O0
102: F3
103: 0
104: 2
105: F3
106: 0
107: 3
108: K0
109: K8
110: O7
111: G5
112: 21
113: 41
114: 32
115: H0
116: O0
117: 13
118: G5
119: 24
120: O0
121: 9
122: 41
123: 16
124: D3

125: P0
126: G9
127: 9
128: D3
129: 18
130: A7
131: G5
132: P1
133: G4
134: P0
135: K8
136: 52
137: G4
138: 22
139: D6
140: K8
141: 3
142: 26
143: O6
144: M2
145: D6
146: K8
147: 3
148: 58
149: L4
150: M2
151: J2
152: 2
153: K8
154: 4
155: A5
156: 15
157: L4
158: M3
159: J2
160: 6
161: K8
162: 4
163: A5
164: N9
165: O6
166: M3
167: E9

168: P1
169: G8
170: I3
171: P3
172: G6
173: 24
174: M4
175: 16
176: O0
177: 15
178: M4
179: 32
180: K8
181: 1
182: 58
183: G6
184: 23
185: M4
186: 48
187: K8
188: 5
189: K8
190: 64
191: 26
192: 26
193: 26
194: F1
195: P3
196: J4
197: 48
198: A0
199: 36

200: G9
201: 80
202: 0
203: E3
204: C5
205: D1
206: C7
207: G9

208: 4
209: 7

210: E3
211: C7
212: D1
213: C7
214: N5
215: H0
216: G0
217: 1
218: 1
219: I3

220: P3
221: J6
222: O8
223: 48
224: 7
225: G5
226: P4
227: J6
228: O8
229: K8

230: 1
231: N5
232: 72
233: 32
234: 70
235: 6
236: A4
237: 74
238: 74
239: 74

```
240: 74
241: 32
242: 70
243:  6
244: D6
245: 16
246: M8
247: M6
248: 48
249: F2
```

```
250: E3
251: D7
252: D5
253: C7
254: A7
255: A7
```

First Payload Demonstration

$000019=Powerup State
$000086=Slippery Level Flag
$000040=Background Transparency
$001493=Beat Level
$001434=Secret Exit
$000071=Player Animation

Copy Jailbreak to Unused RAM

```
LDX #$00         A2 00
LDA $FFFE00,x    BF 00 FE FF
STA $7F8200,x    9F 00 82 7F
DEX              CA
BNE -            D0 F5
LDA #$7F         A9 7F
STA $FF          85 FF
```

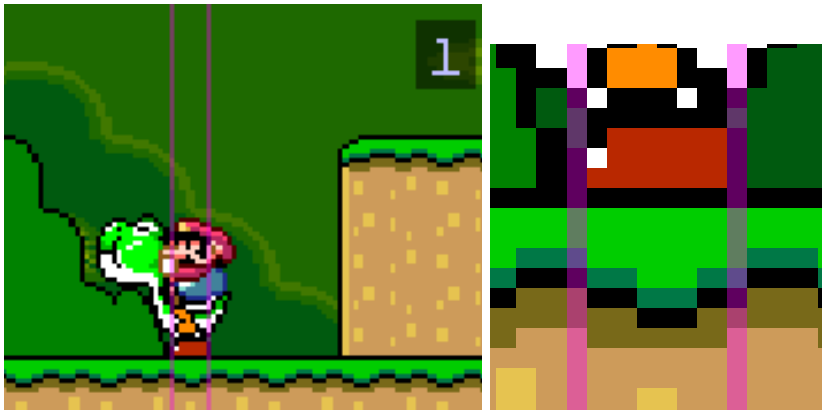
```
LDA #$82          A9 82
STA $FE           85 FE
RTL               6B
```

RLX ACE to Install Jailbreak

Unplug both multitaps, and plug the gameplay controller into port 1 of the console, and the L, down, select, Y, B controller into port 2 of the console. Reclip it to L, down, right, select, Y, B.

Without powering off, remove the jailbroken cartridge from the SNES, and plug in the fresh cartridge. Press the reset switch. Delete File C, and play on File A.

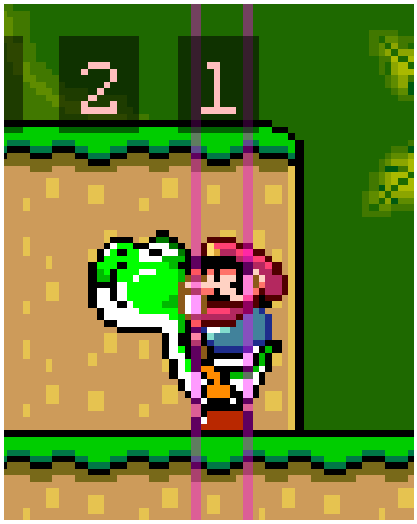
FC



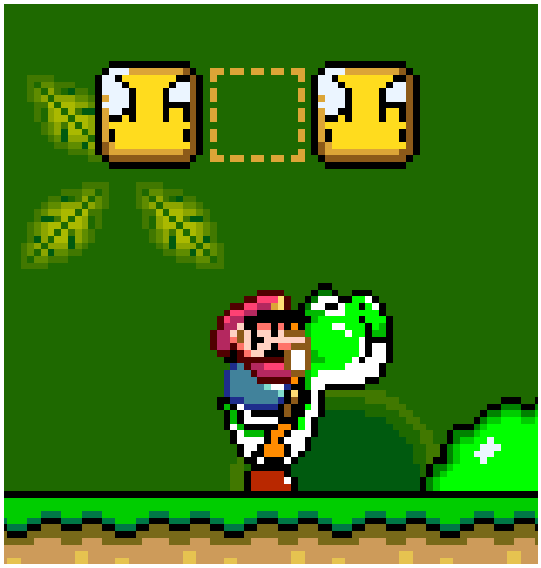
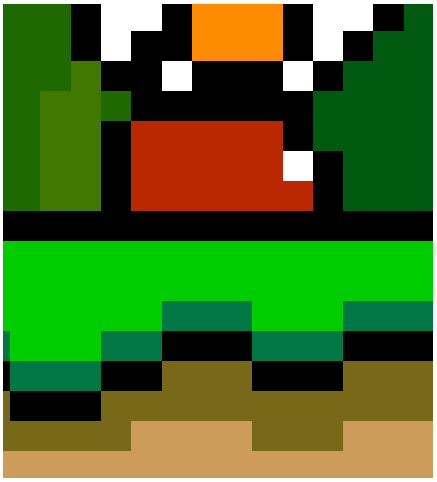
5C, beginning of the level

Also 5C, but farther right

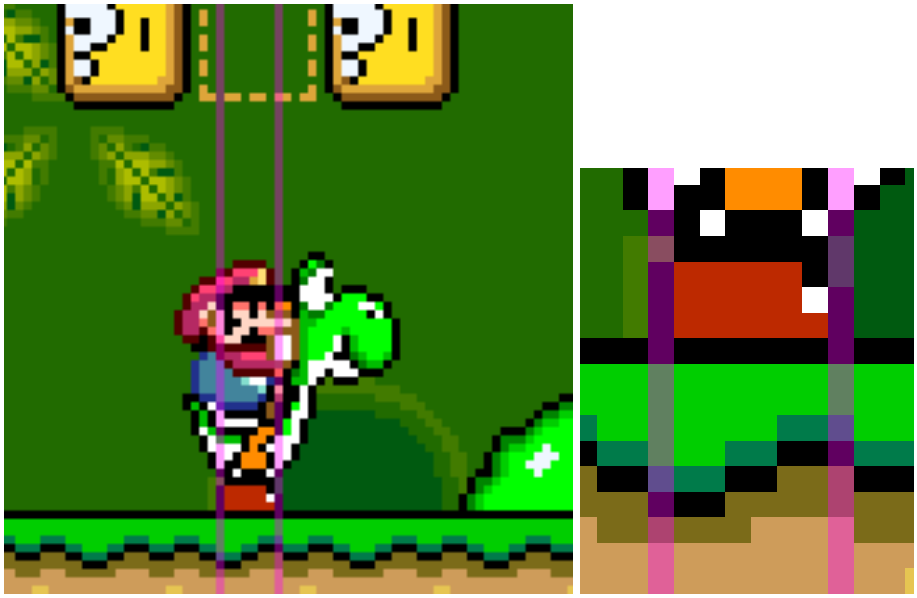

0D




82

7F



Second Payload Demonstration

```
A4 F0 A5 18 29 10 F0 10 AA 88 10 02 A0 0B 84 F0
B9 C8 14 D0 03 CA 10 F1 B9 A0 15 3A 39 C8 14 F0
1F 98 18 69 0C A8 F4 40 03 A2 06 20 62 FF 68 1A
48 A2 08 A4 F0 20 62 FF 68 68 9E 70 04 CA 10 FA
24 17 50 1D A6 F0 A5 15 29 03 A8 B9 CA 8A 50 09
95 B6 A5 15 4A 4A B8 80 EF 95 AA 22 2A 80 01 64
15 6B 08 B9 D4 14 EB B9 D8 00 C2 20 F5 CB 4A 4A
4A 28 48 B5 78 69 08 A0 20 93 04 63 01 48 A9 2D
99 42 03 A9 F4 99 43 03 68 88 88 88 88 10 EA 68
60
```