

To decode an NES Game Genie code, first translate each character into its hexadecimal equivalent using the following table:

A	0x0
P	0x1
Z	0x2
L	0x3
G	0x4
I	0x5
T	0x6
Y	0x7
E	0x8
O	0x9
X	0xA
U	0xB
K	0xC
S	0xD
V	0xE
N	0xF

Then comes the tricky part. There is a lot of convoluted bit shifting that occurs in order to get the address and data from a Game Genie. This is probably to make the Game Genie codes seem more magical. After all, given 2 Game Genie codes, one that granted 5 lives on startup and another code that granted 9 lives, and the only difference between the 2 codes was one character, even a novice player could probably figure out that modifying that one character to any of the acceptable letter characters would grant between 1 and 16 lives on startup.

In order to decode the 6-character NES Game Genie code, name the 6 characters/hex values [n0 .. n5]. Follow the pseudocode (which assumes you know the C-style operators for bitwise AND (&), bitwise OR (|), and the << and >> bit shift operators):

```
address = 0x8000 +
    ((n3 & 7) << 12)
    | ((n5 & 7) << 8) | ((n4 & 8) << 8)
    | ((n2 & 7) << 4) | ((n1 & 8) << 4)
    | (n4 & 7)    | (n3 & 8);
```

The algorithm simply combines the lower 3 bits from one 4-bit nibble and the top bit from another nibble and puts the resulting nibble somewhere else. Here is the data algorithm:

```
data =
    ((n1 & 7) << 4) | ((n0 & 8) << 4)
    | (n0 & 7)    | (n5 & 8);
```

Example: The code is GOSSIP (amazing coincidence that it happens to also be an English word). This works in Capcom's Ghosts 'n Goblins to start your player with a really funky weapon. Work through the code by hand to see if understand the decoding algorithm.

```

n0 n1 n2 n3 n4 n5
G O S S I P
0x4 0x9 0xD 0xD 0x5 0x1
0100 1001 1101 1101 0101 0001

```

address = 0xD1DD, data = 0x14

Therefore, whenever the CPU reads from address 0xD1DD, the Game Genie will intercept the read and return the byte 0x14.

Since the code is scrambled, you will need the table below to descramble it.

Char #	1	2	3	4	5	6	
Bit #	3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0 3 2 1 0						
maps to	1 6 7 8 H 2 3 4	- I J K L A B C D M N O 5 E F G					

Game Genie Codes

APUXLZIA 10 minutes per quarter instead of 5
PUXLZIA 15 minutes per quarter
AZUXLZIA 20 minutes per quarter
ZAUXLZIA 2 minutes per quarter
SXNXPZVG Freeze timer for continuous play (Press RESET to stop)
AAOATTTA Touchdown scores 0 instead of 6,player 1
AEOEVITA Touchdown scores 0,player 2 or computer
LAOATTTA Touchdown scores 3,player 1
LEOEVITA Touchdown scores 3,player 2 or computer
PAOATTTTE Touchdown scores 9,player 1
PEOEVITE Touchdown scores 9,player 2 or computer
GAOATTTTE Touchdown scores 12,player 1
GEOEVITE Touchdown scores 12,player 2 or computer
AAEALYPA Extra-point kick scores 0 instead of 1,player 1
AEEEUTPA Extra-point kick scores 0,player 2 or computer
ZAEALYPA Extra-point kick scores 2,player 1
ZEEEUTPA Extra-point kick scores 2,player 2 or computer
LAEALYPA Extra-point kick scores 3,player 1
LEEEUTPA Extra-point kick scores 3,player 2 or computer
TAEALYPA Extra-point kick scores 6,player 1
TEEEUTPA Extra-point kick scores 6,player 2 or computer

AEKAGGLA Field goal scores 0 instead of 3,player 1
 AAKEKGLA Field goal scores 0,player 2 or computer
 PEKAGGLA Field goal scores 1,player 1
 PAKEKGLA Field goal scores 1,player 2 or computer
 TEKAGGLA Field goal scores 6,player 1
 TAKEKGLA Field goal scores 6,player 2 or computer
 PEKAGGLE Field goal scores 9,player 1
 PAKEKGLE Field goal scores 9,player 2 or computer
 AASASIZA Safety scores 0 instead of 2,player 1
 AEKEIIZA Safety scores 0,player 2 or computer
 PASASIZA Safety scores 1,player 1
 PEKEIIZA Safety scores 1,player 2 or computer
 GASASIZA Safety scores 4,player 1
 GEKEIIZA Safety scores 4,player 2 or computer
 TASASIZA Safety scores 6,player 1
 TEKEIIZA Safety scores 6,player 2 or computer

8-Character Codes

8-character NES Game Genie codes are similar to the 6-character variety except that there is also a compare byte value that needs to be decoded as well. This is most likely due to the fact that many games use memory mappers in order to increase the amount of code and data they can use. Since the game might be swapping program (PRG) banks in and out of the CPU address space, the Game Genie can't just return the code data value when the CPU reads from a particular address. Instead, when the CPU reads from the cartridge, the Game Genie checks if the address in the cartridge equals the compare value and returns the code data value if it does; otherwise, it returns the real value in the cartridge.

The algorithm for decoding the address of an 8-character code is the same as decoding the address for a 6-character code. The algorithm for decoding the data byte changes a little:

```

data =
  ((n1 & 7) << 4) | ((n0 & 8) << 4)
  | (n0 & 7)      | (n7 & 8);

```

And the algorithm for decoding the compare value is as follows:

```

compare =
  ((n7 & 7) << 4) | ((n6 & 8) << 4)

```

| (n6 & 7) | (n5 & 8);

Example: The code is ZEXPYGLA. This works on Dr. Mario in order to clear a row or column with only 3 colors in a line, rather than 4. Work through the code by hand to see if understand the decoding algorithm.

```
n0 n1 n2 n3 n4 n5 n6 n7
Z E X P Y G L A
0x2 0x8 0xA 0x1 0x7 0x4 0x3 0x0
0010 1000 1010 0001 0111 0100 0011 0000
```

address = 0x94A7, data = 0x02, compare = 0x03

Therefore, when the CPU reads from address 0x94A7 and the real byte at that address is 0x03, then the Game Genie will return 0x02 instead of 0x03. If the byte is something other than 0x03, then that byte will be returned.