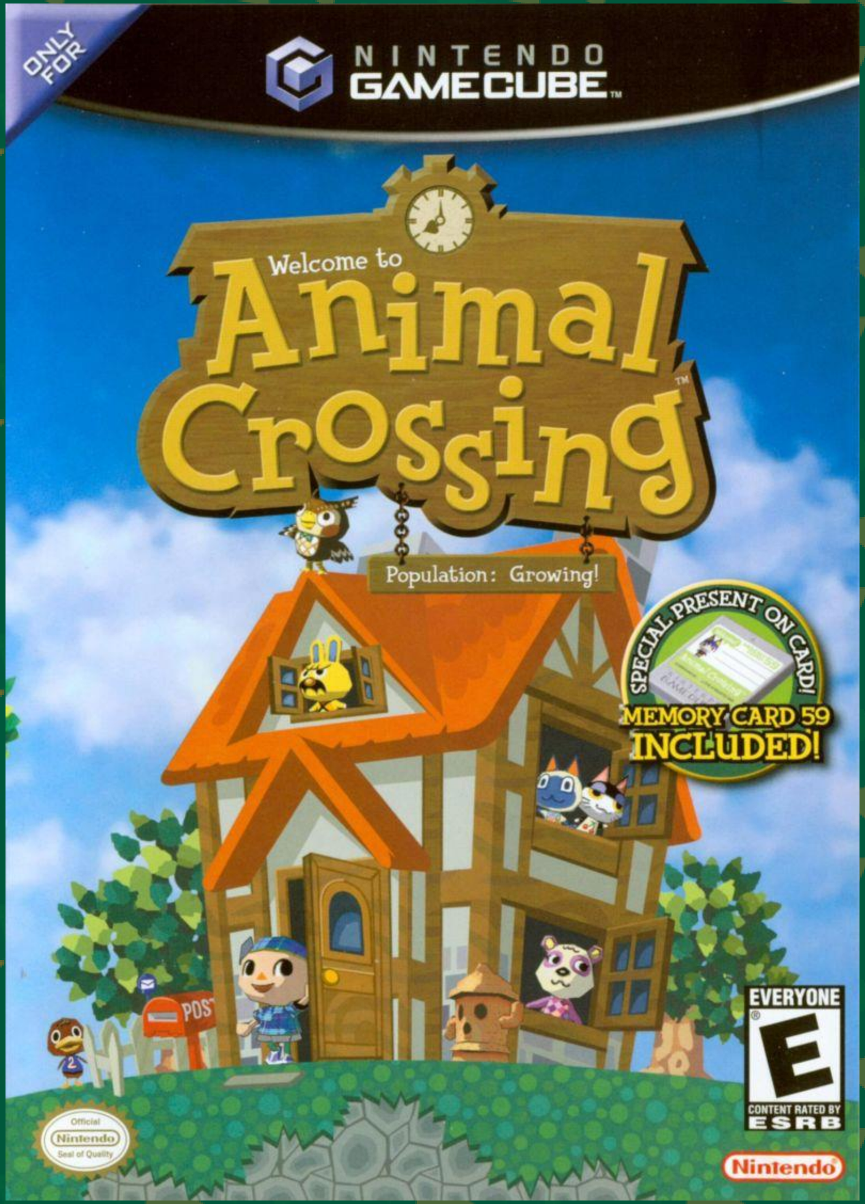
Secrets of Animal Crossing

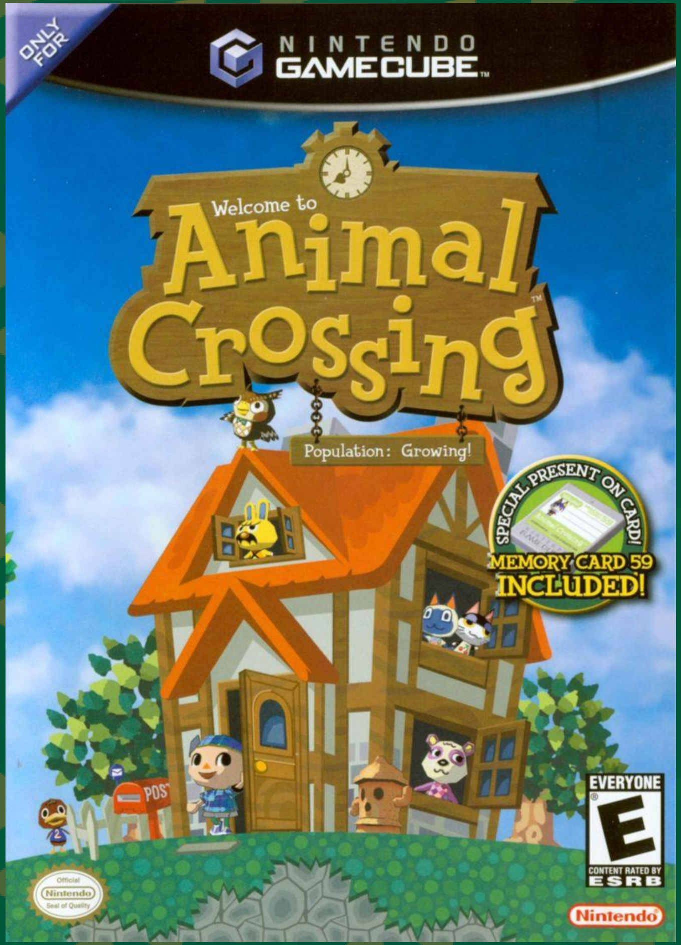
Adventures in ROM Hacking

James Chambers

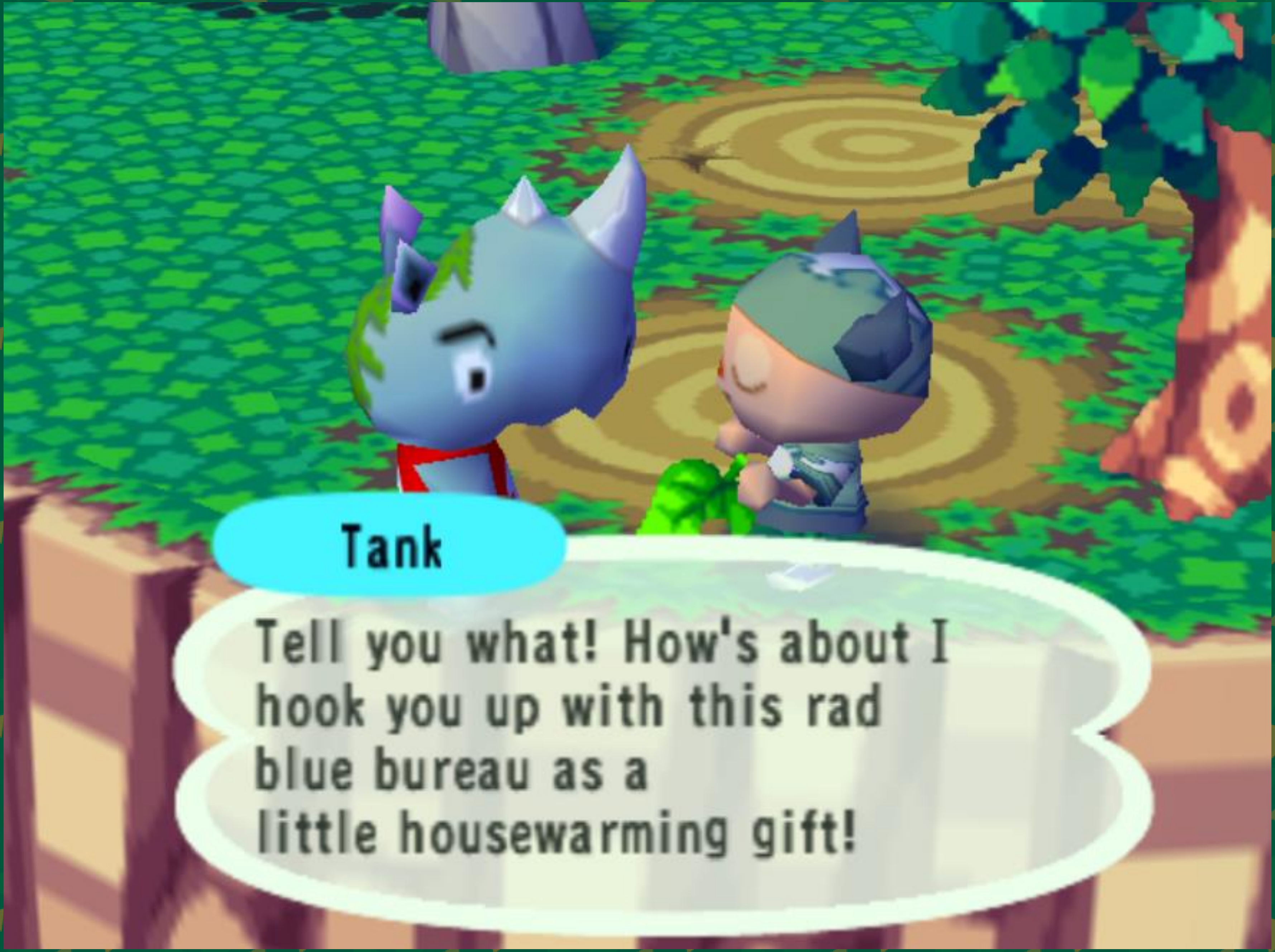

ONLY FOR
NINTENDO
GAMECUBE
Welcome to
Animal Crossing
Population: Growing!
SPECIAL PRESENT ON CARD!
MEMORY CARD 59 INCLUDED!
POST
EVERYONE
E
CONTENT RATED BY
ESRB
Official Nintendo Seal of Quality
Nintendo

Introduction

- Animal Crossing
  - Released in 2002
- “Life simulator” game
  - Interact with villagers
  - Customize house, collect items
  - Events and holidays happen in real time, whether or not you’re playing

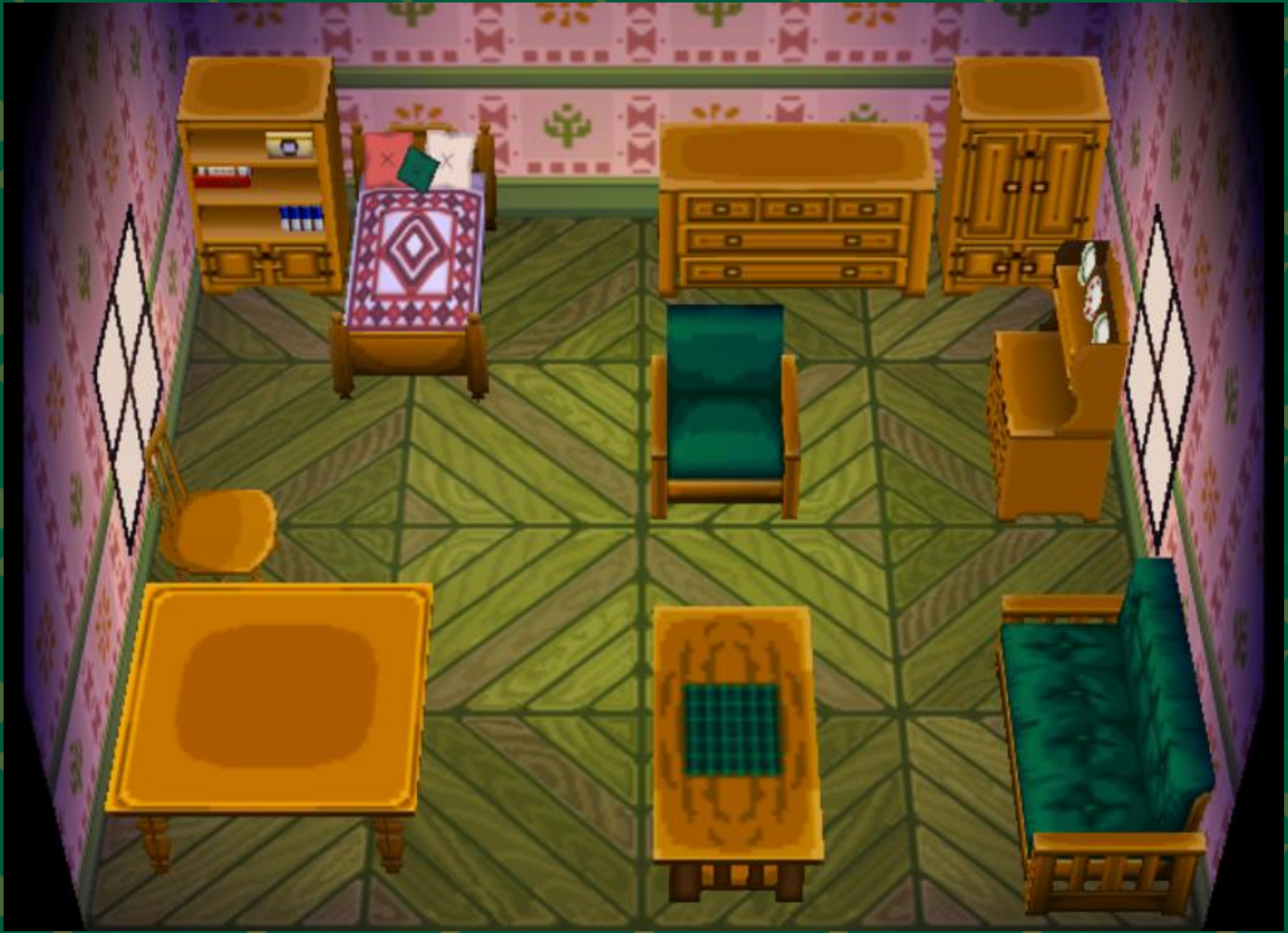


Nook's Cranny
5 . 19 Sat
3:49 p.m.



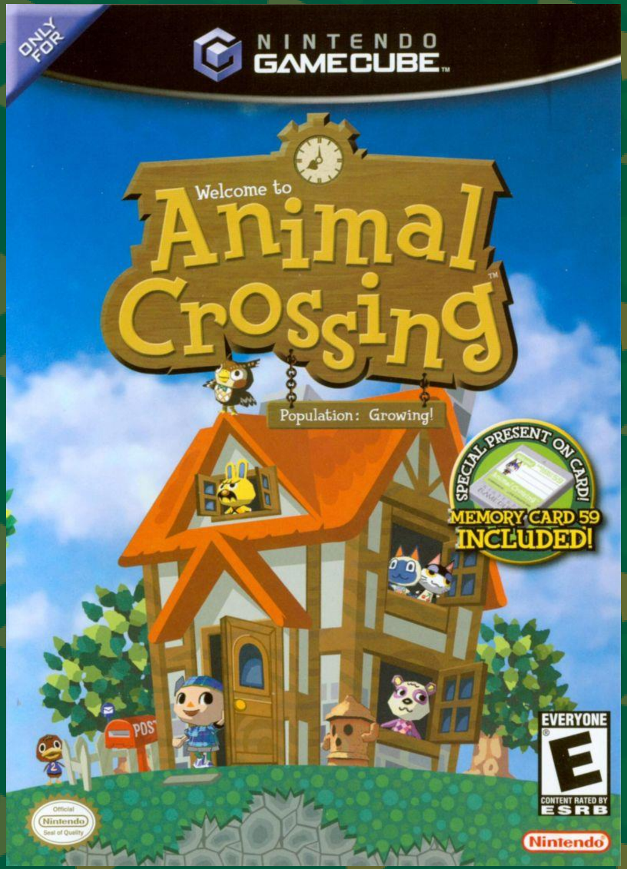
Tank

Tell you what! How's about I hook you up with this rad blue bureau as a little housewarming gift!



PERICARD
NINTENDO64
どうぶつの森
ひとりより ふたり
ふたりより よにん
よにんより… たーくさん。
ひとりよりふたり
コミュニケーションゲーム
Nintendo
コントローラパック対応 121
バックアップ機能付き

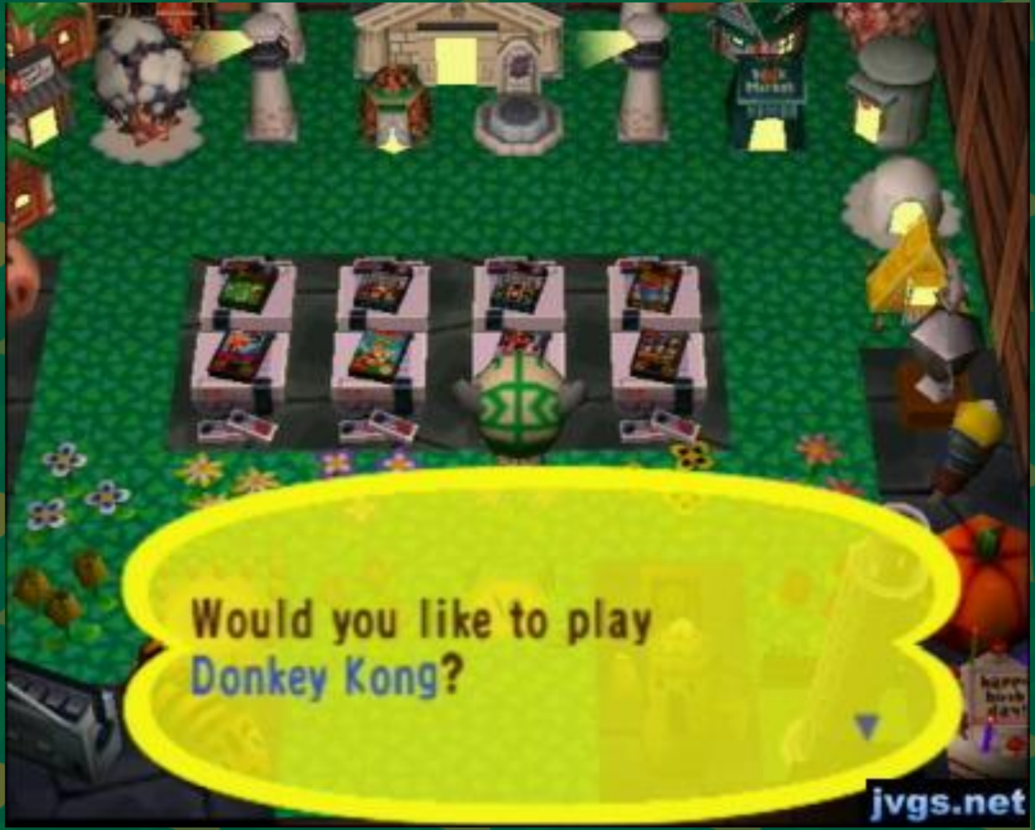
Nintendo
NINTENDO GAMECUBE
おまけデータ入り
メモリーカード59が付いてるよ！
どうぶつの森+
ひとりより ふたり
ふたりより よにん
よにんより… たーくさん。
ぷらすになって
いろいろぷらす
ひとりよりふたり
ゲームボーイ アドバンスとつないで
もっと も～っと、いいくらし。
■コミュニケーション
メモリーカード使用ブロック数 59
ゲームボーイアドバンスとの接続には GBAケーブル（別売）が必要です。


ONLY FOR
NINTENDO GAMECUBE
Welcome to
Animal Crossing
Population: Growing!
SPECIAL PRESENT ON CARD!
MEMORY CARD 59 INCLUDED!
POST
EVERYONE
E
CONTENT RATED BY ESRB
Official Nintendo Seal of Quality
Nintendo

Nintendo
NINTENDO GAMECUBE
カードeリーダー+・カードe+付き
どうぶつの森e+
もっと、いい！ぷらす
DOUBUTSU NO MORI e+
つながる
CERO 全年齢
■1～4人用
■コミュニケーション
メモリーカード使用ブロック数 59～

Introduction

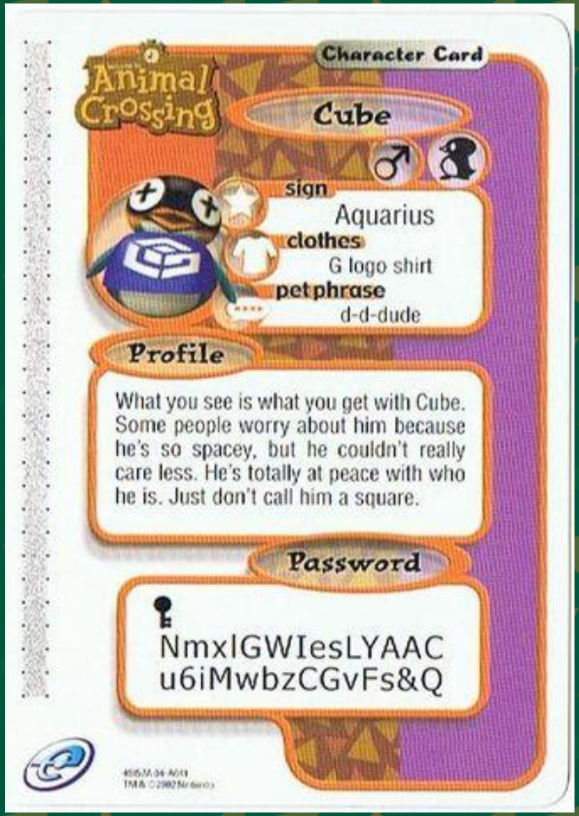
- Does N64 emulation (graphics, some system properties) for main game
- NES emulator for NES games you can acquire in-world



Would you like to play Donkey Kong?

Introduction

- Connects to Game Boy Advance
  - Can transmit NES games to run on Gameboy Advance
  - E-Reader peripheral for collectible cards that grant items, etc. (like an early version of Amiibo)
  - Unlocks game features



Animal Crossing

Character Card

Cube

♂

sign
Aquarius

clothes
G logo shirt

pet phrase
d-d-dude

Profile

What you see is what you get with Cube. Some people worry about him because he's so spacey, but he couldn't really care less. He's totally at peace with who he is. Just don't call him a square.

Password

NmxIGWIesLYAAC
u6iMwbzCGvFs&Q

4505A-04-A013
TM & ©2002 Nintendo

Introduction

- GameCube contains customized PowerPC processor
  - Extended instruction set

Introduction

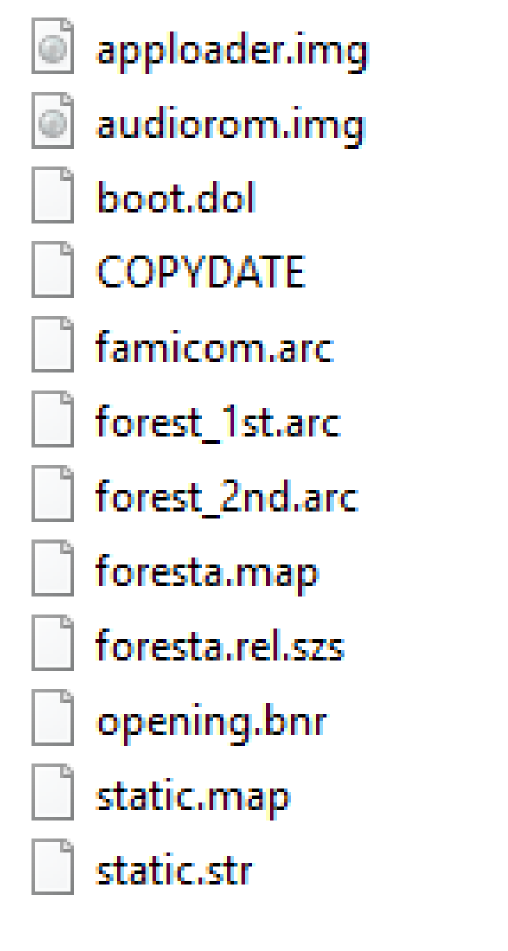
- Halloween was approaching and it’d be fun to make a spooky mod
  - Lack of tutorials on doing comprehensive ROM hacking
  - I could make a tutorial in the style that I learned RE/cracking from
- Goals for mod:
  - Create a new holiday based event
- Targets:
  - Dialogue system
  - Event system
  - Quest system

Looking Inside

File formats, symbols, IDA scripts

Looking Inside

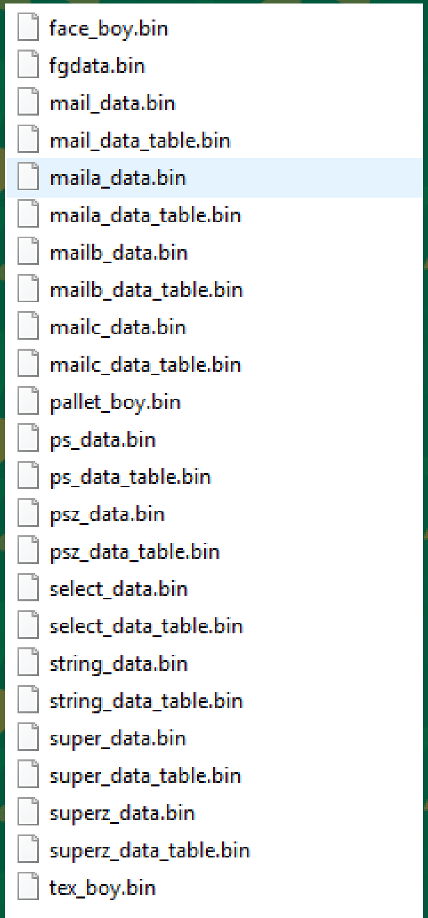
- Open up the disc image:
  - boot.dol
  - foresta.rel.szs
  - forest_1st.arc
  - forest_2nd.arc
  - famicom.arc
  - statica.map
  - foresta.map
- Lots of proprietary formats
  - File format analysis will be important
  - Some documentation on common GameCube/Wii formats already exists



apploader.img
audiorom.img
boot.dol
COPYDATE
famicom.arc
forest_1st.arc
forest_2nd.arc
foresta.map
foresta.rel.szs
opening.bnr
static.map
static.str

Looking Inside

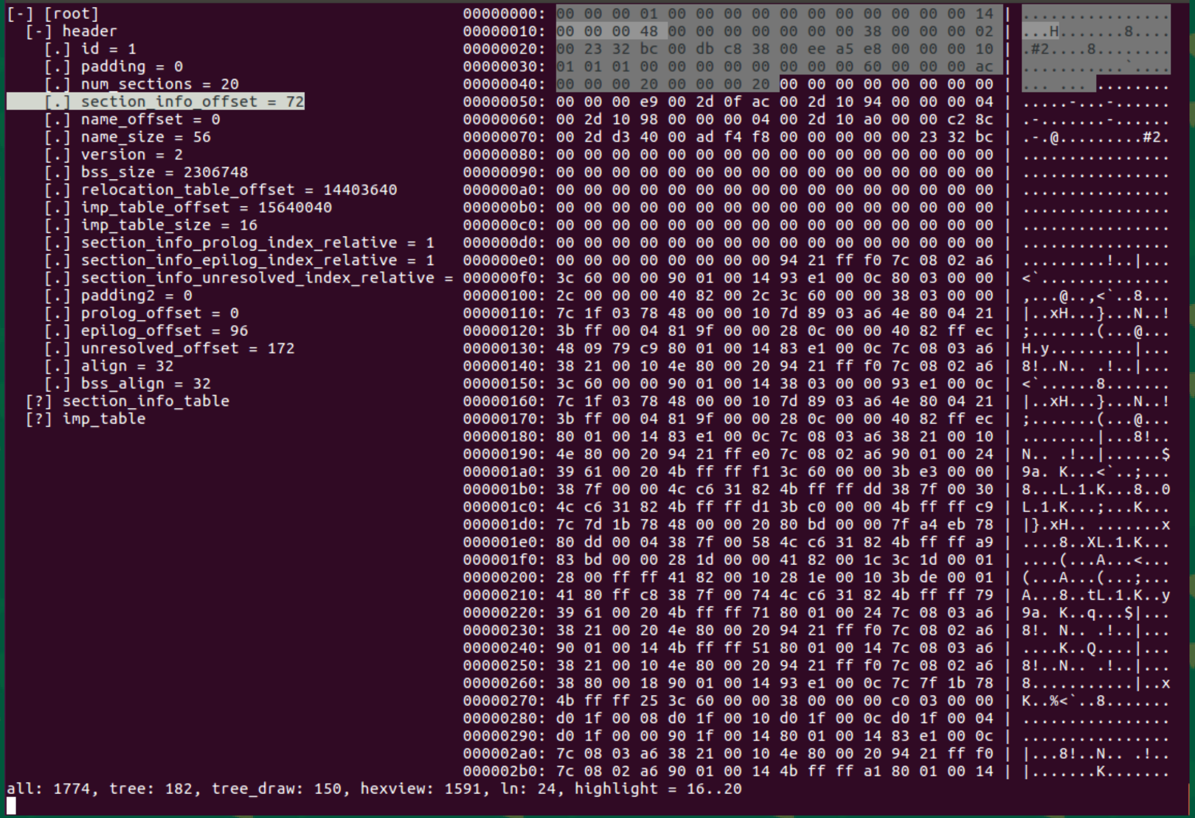
- .ARC files: archives
  - Contain most of the interesting data files
- Some tools for opening ARC files but not creating them (except maybe sketchy EXEs)
- Found a Python extractor and added archive creation to it
  - Noticed why the other tools didn't support it
  - It'd be nice to have some generic tools for defining/analyzing binary formats
    - Kaitai and the other thing for game archives



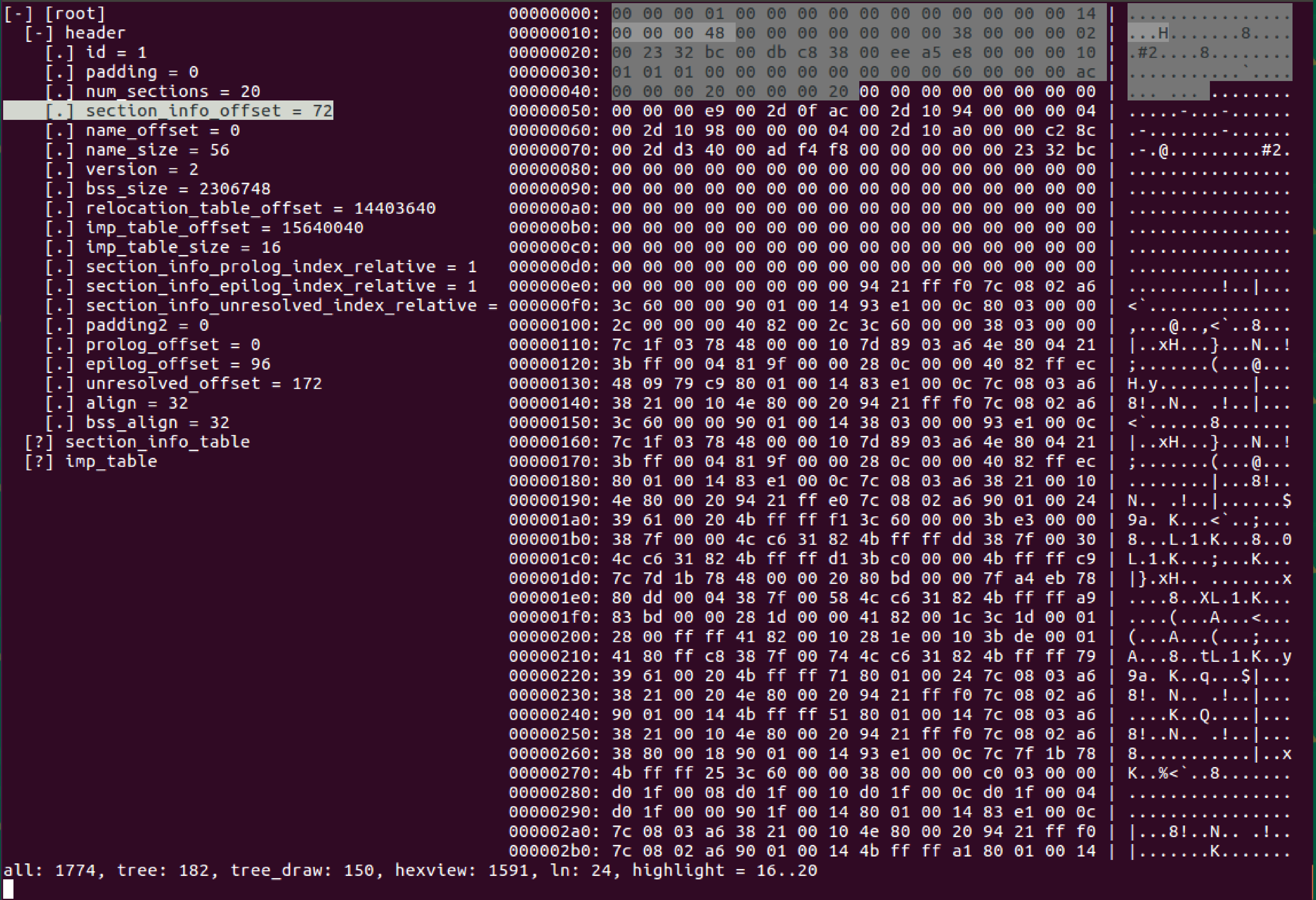
```
face_boy.bin
fgdata.bin
mail_data.bin
mail_data_table.bin
maila_data.bin
maila_data_table.bin
mailb_data.bin
mailb_data_table.bin
mailc_data.bin
mailc_data_table.bin
pallet_boy.bin
ps_data.bin
ps_data_table.bin
psz_data.bin
psz_data_table.bin
select_data.bin
select_data_table.bin
string_data.bin
string_data_table.bin
super_data.bin
super_data_table.bin
superz_data.bin
superz_data_table.bin
tex_boy.bin
```

Looking Inside

- Binary files
  - boot.dol
  - foresta.rel
- Importing to IDA
  - Custom PPC instructions
    - *“Paired singles are a unique part of the Gekko/Broadway processors used in the Gamecube and Wii. They provide fast vector math by keeping two single-precision floating point numbers in a single floating point register, and doing math across registers.”*
    - PPC Altivec IDA plugin: https://github.com/nihilus/PPCAltivec
  - Custom REL/DOL loaders
    - https://github.com/heinermann/ida-wii-loaders
- Kaitai definitions for debugging loaders



```
[-] [root]                                       00000000: 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 14 | ................
  [-] header                                     00000010: 00 00 00 48 00 00 00 00 00 00 00 38 00 00 00 02 | ...H.......8....
    [.] id = 1                                   00000020: 00 23 32 bc 00 db c8 38 00 ee a5 e8 00 00 00 10 | .#2....8........
    [.] padding = 0                              00000030: 01 01 01 00 00 00 00 00 00 00 00 60 00 00 00 ac | ...........`....
    [.] num_sections = 20                        00000040: 00 00 00 20 00 00 00 20 00 00 00 00 00 00 00 00 | ... ... ........
    [.] section_info_offset = 72                 00000050: 00 00 00 e9 00 2d 0f ac 00 2d 10 94 00 00 00 04 | .....-...-......
    [.] name_offset = 0                          00000060: 00 2d 10 98 00 00 00 04 00 2d 10 a0 00 00 c2 8c | .-.......-......
    [.] name_size = 56                           00000070: 00 2d d3 40 00 ad f4 f8 00 00 00 00 00 23 32 bc | .-.@.........#2.
    [.] version = 2                              00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] bss_size = 2306748                       00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] relocation_table_offset = 14403640       000000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] imp_table_offset = 15640040              000000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] imp_table_size = 16                      000000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] section_info_prolog_index_relative = 1   000000d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] section_info_epilog_index_relative = 1   000000e0: 00 00 00 00 00 00 00 00 94 21 ff f0 7c 08 02 a6 | .........!..|...
    [.] section_info_unresolved_index_relative = 000000f0: 3c 60 00 00 90 01 00 14 93 e1 00 0c 80 03 00 00 | <`..............
    [.] padding2 = 0                             00000100: 2c 00 00 00 40 82 00 2c 3c 60 00 00 38 03 00 00 | ,...@..,<`..8...
    [.] prolog_offset = 0                        00000110: 7c 1f 03 78 48 00 00 10 7d 89 03 a6 4e 80 04 21 | |..xH...}...N..!
    [.] epilog_offset = 96                       00000120: 3b ff 00 04 81 9f 00 00 28 0c 00 00 40 82 ff ec | ;.......(...@...
    [.] unresolved_offset = 172                  00000130: 48 09 79 c9 80 01 00 14 83 e1 00 0c 7c 08 03 a6 | H.y.........|...
    [.] align = 32                               00000140: 38 21 00 10 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!..N.. .!..|...
    [.] bss_align = 32                           00000150: 3c 60 00 00 90 01 00 14 38 03 00 00 93 e1 00 0c | <`......8.......
  [?] section_info_table                         00000160: 7c 1f 03 78 48 00 00 10 7d 89 03 a6 4e 80 04 21 | |..xH...}...N..!
  [?] imp_table                                  00000170: 3b ff 00 04 81 9f 00 00 28 0c 00 00 40 82 ff ec | ;.......(...@...
                                                 00000180: 80 01 00 14 83 e1 00 0c 7c 08 03 a6 38 21 00 10 | ........|...8!..
                                                 00000190: 4e 80 00 20 94 21 ff e0 7c 08 02 a6 90 01 00 24 | N.. .!..|......$
                                                 000001a0: 39 61 00 20 4b ff ff f1 3c 60 00 00 3b e3 00 00 | 9a. K...<`..;...
                                                 000001b0: 38 7f 00 00 4c c6 31 82 4b ff ff dd 38 7f 00 30 | 8...L.1.K...8..0
                                                 000001c0: 4c c6 31 82 4b ff ff d1 3b c0 00 00 4b ff ff c9 | L.1.K...;...K...
                                                 000001d0: 7c 7d 1b 78 48 00 00 20 80 bd 00 00 7f a4 eb 78 | |}.xH.. .......x
                                                 000001e0: 80 dd 00 04 38 7f 00 58 4c c6 31 82 4b ff ff a9 | ....8..XL.1.K...
                                                 000001f0: 83 bd 00 00 28 1d 00 00 41 82 00 1c 3c 1d 00 01 | ....(...A...<...
                                                 00000200: 28 00 ff ff 41 82 00 10 28 1e 00 10 3b de 00 01 | (...A...(...;...
                                                 00000210: 41 80 ff c8 38 7f 00 74 4c c6 31 82 4b ff ff 79 | A...8..tL.1.K..y
                                                 00000220: 39 61 00 20 4b ff ff 71 80 01 00 24 7c 08 03 a6 | 9a. K..q...$|...
                                                 00000230: 38 21 00 20 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!. N.. .!..|...
                                                 00000240: 90 01 00 14 4b ff ff 51 80 01 00 14 7c 08 03 a6 | ....K..Q....|...
                                                 00000250: 38 21 00 10 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!..N.. .!..|...
                                                 00000260: 38 80 00 18 90 01 00 14 93 e1 00 0c 7c 7f 1b 78 | 8...........|..x
                                                 00000270: 4b ff ff 25 3c 60 00 00 38 00 00 00 c0 03 00 00 | K..%<`..8.......
                                                 00000280: d0 1f 00 08 d0 1f 00 10 d0 1f 00 0c d0 1f 00 04 | ................
                                                 00000290: d0 1f 00 00 90 1f 00 14 80 01 00 14 83 e1 00 0c | ................
                                                 000002a0: 7c 08 03 a6 38 21 00 10 4e 80 00 20 94 21 ff f0 | |...8!..N.. .!..
                                                 000002b0: 7c 08 02 a6 90 01 00 14 4b ff ff a1 80 01 00 14 | |.......K.......
all: 1774, tree: 182, tree_draw: 150, hexview: 1591, ln: 24, highlight = 16..20
```



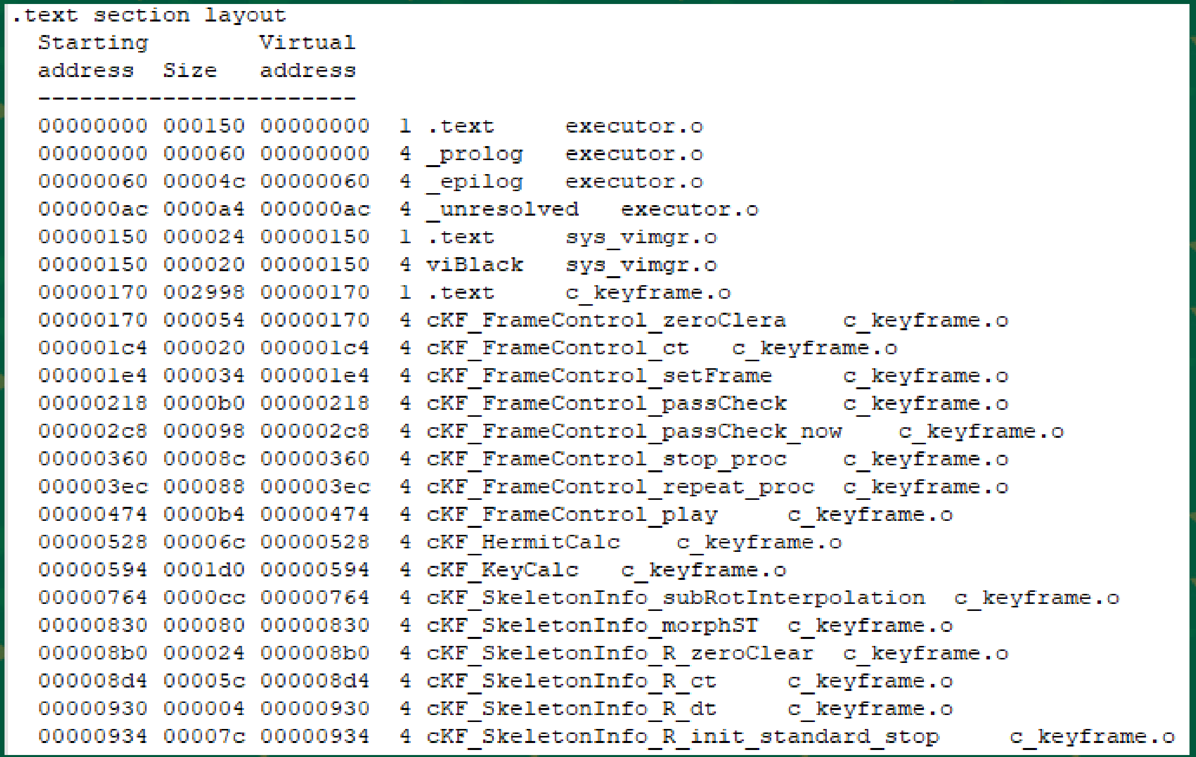
```
[-] [root]                                       00000000: 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 14 | ................
  [-] header                                     00000010: 00 00 00 48 00 00 00 00 00 00 00 38 00 00 00 02 | ...H.......8....
    [.] id = 1                                   00000020: 00 23 32 bc 00 db c8 38 00 ee a5 e8 00 00 00 10 | .#2....8........
    [.] padding = 0                              00000030: 01 01 01 00 00 00 00 00 00 00 00 60 00 00 00 ac | ...........`....
    [.] num_sections = 20                        00000040: 00 00 00 20 00 00 00 20 00 00 00 00 00 00 00 00 | ... ... ........
    [.] section_info_offset = 72                 00000050: 00 00 00 e9 00 2d 0f ac 00 2d 10 94 00 00 00 04 | .....-...-......
    [.] name_offset = 0                          00000060: 00 2d 10 98 00 00 00 04 00 2d 10 a0 00 00 c2 8c | .-.......-......
    [.] name_size = 56                           00000070: 00 2d d3 40 00 ad f4 f8 00 00 00 00 00 23 32 bc | .-.@.........#2.
    [.] version = 2                              00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] bss_size = 2306748                       00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] relocation_table_offset = 14403640       000000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] imp_table_offset = 15640040              000000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] imp_table_size = 16                      000000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] section_info_prolog_index_relative = 1   000000d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
    [.] section_info_epilog_index_relative = 1   000000e0: 00 00 00 00 00 00 00 00 94 21 ff f0 7c 08 02 a6 | .........!..|...
    [.] section_info_unresolved_index_relative = 000000f0: 3c 60 00 00 90 01 00 14 93 e1 00 0c 80 03 00 00 | <`..............
    [.] padding2 = 0                             00000100: 2c 00 00 00 40 82 00 2c 3c 60 00 00 38 03 00 00 | ,...@..,<`..8...
    [.] prolog_offset = 0                        00000110: 7c 1f 03 78 48 00 00 10 7d 89 03 a6 4e 80 04 21 | |..xH...}...N..!
    [.] epilog_offset = 96                       00000120: 3b ff 00 04 81 9f 00 00 28 0c 00 00 40 82 ff ec | ;.......(...@...
    [.] unresolved_offset = 172                  00000130: 48 09 79 c9 80 01 00 14 83 e1 00 0c 7c 08 03 a6 | H.y.........|...
    [.] align = 32                               00000140: 38 21 00 10 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!..N.. .!..|...
    [.] bss_align = 32                           00000150: 3c 60 00 00 90 01 00 14 38 03 00 00 93 e1 00 0c | <`......8.......
  [?] section_info_table                         00000160: 7c 1f 03 78 48 00 00 10 7d 89 03 a6 4e 80 04 21 | |..xH...}...N..!
  [?] imp_table                                  00000170: 3b ff 00 04 81 9f 00 00 28 0c 00 00 40 82 ff ec | ;.......(...@...
                                                 00000180: 80 01 00 14 83 e1 00 0c 7c 08 03 a6 38 21 00 10 | ........|...8!..
                                                 00000190: 4e 80 00 20 94 21 ff e0 7c 08 02 a6 90 01 00 24 | N.. .!..|......$
                                                 000001a0: 39 61 00 20 4b ff ff f1 3c 60 00 00 3b e3 00 00 | 9a. K...<`..;...
                                                 000001b0: 38 7f 00 00 4c c6 31 82 4b ff ff dd 38 7f 00 30 | 8...L.1.K...8..0
                                                 000001c0: 4c c6 31 82 4b ff ff d1 3b c0 00 00 4b ff ff c9 | L.1.K...;...K...
                                                 000001d0: 7c 7d 1b 78 48 00 00 20 80 bd 00 00 7f a4 eb 78 | |}.xH.. .......x
                                                 000001e0: 80 dd 00 04 38 7f 00 58 4c c6 31 82 4b ff ff a9 | ....8..XL.1.K...
                                                 000001f0: 83 bd 00 00 28 1d 00 00 41 82 00 1c 3c 1d 00 01 | ....(...A...<...
                                                 00000200: 28 00 ff ff 41 82 00 10 28 1e 00 10 3b de 00 01 | (...A...(...;...
                                                 00000210: 41 80 ff c8 38 7f 00 74 4c c6 31 82 4b ff ff 79 | A...8..tL.1.K..y
                                                 00000220: 39 61 00 20 4b ff ff 71 80 01 00 24 7c 08 03 a6 | 9a. K..q...$|...
                                                 00000230: 38 21 00 20 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!. N.. .!..|...
                                                 00000240: 90 01 00 14 4b ff ff 51 80 01 00 14 7c 08 03 a6 | ....K..Q....|...
                                                 00000250: 38 21 00 10 4e 80 00 20 94 21 ff f0 7c 08 02 a6 | 8!..N.. .!..|...
                                                 00000260: 38 80 00 18 90 01 00 14 93 e1 00 0c 7c 7f 1b 78 | 8...........|..x
                                                 00000270: 4b ff ff 25 3c 60 00 00 38 00 00 00 c0 03 00 00 | K..%<`..8.......
                                                 00000280: d0 1f 00 08 d0 1f 00 10 d0 1f 00 0c d0 1f 00 04 | ................
                                                 00000290: d0 1f 00 00 90 1f 00 14 80 01 00 14 83 e1 00 0c | ................
                                                 000002a0: 7c 08 03 a6 38 21 00 10 4e 80 00 20 94 21 ff f0 | |...8!..N.. .!..
                                                 000002b0: 7c 08 02 a6 90 01 00 14 4b ff ff a1 80 01 00 14 | |.......K.......
all: 1774, tree: 182, tree_draw: 150, hexview: 1591, ln: 24, highlight = 16..20
```

Looking Inside

- The symbol map files
  - GameCube build script options:

```
# -map                         - create a
.MAP file that shows final memory
layout of  all sections
```

- Make simple IDA script for populating database with names

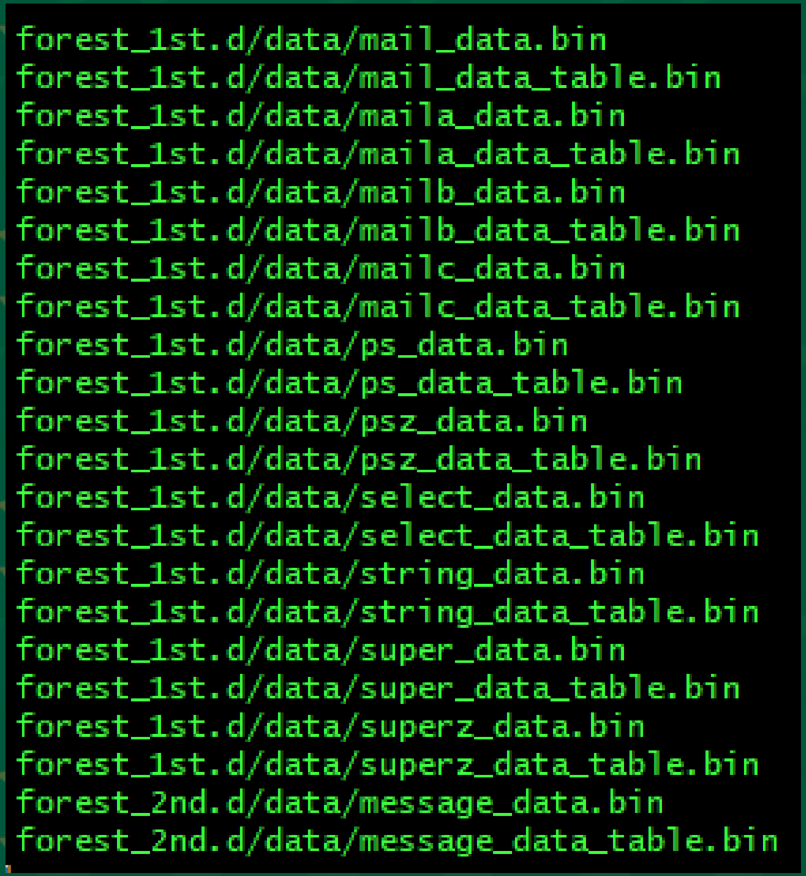


```
.text section layout
  Starting        Virtual
  address  Size   address
  -----------------------
  00000000 000150 00000000  1 .text     executor.o
  00000000 000060 00000000  4 _prolog   executor.o
  00000060 00004c 00000060  4 _epilog   executor.o
  000000ac 0000a4 000000ac  4 _unresolved   executor.o
  00000150 000024 00000150  1 .text     sys_vimgr.o
  00000150 000020 00000150  4 viBlack   sys_vimgr.o
  00000170 002998 00000170  1 .text     c_keyframe.o
  00000170 000054 00000170  4 cKF_FrameControl_zeroClera    c_keyframe.o
  000001c4 000020 000001c4  4 cKF_FrameControl_ct   c_keyframe.o
  000001e4 000034 000001e4  4 cKF_FrameControl_setFrame     c_keyframe.o
  00000218 0000b0 00000218  4 cKF_FrameControl_passCheck    c_keyframe.o
  000002c8 000098 000002c8  4 cKF_FrameControl_passCheck_now    c_keyframe.o
  00000360 00008c 00000360  4 cKF_FrameControl_stop_proc    c_keyframe.o
  000003ec 000088 000003ec  4 cKF_FrameControl_repeat_proc  c_keyframe.o
  00000474 0000b4 00000474  4 cKF_FrameControl_play     c_keyframe.o
  00000528 00006c 00000528  4 cKF_HermitCalc    c_keyframe.o
  00000594 0001d0 00000594  4 cKF_KeyCalc   c_keyframe.o
  00000764 0000cc 00000764  4 cKF_SkeletonInfo_subRotInterpolation  c_keyframe.o
  00000830 000080 00000830  4 cKF_SkeletonInfo_morphST  c_keyframe.o
  000008b0 000024 000008b0  4 cKF_SkeletonInfo_R_zeroClear  c_keyframe.o
  000008d4 00005c 000008d4  4 cKF_SkeletonInfo_R_ct     c_keyframe.o
  00000930 000004 00000930  4 cKF_SkeletonInfo_R_dt     c_keyframe.o
  00000934 00007c 00000934  4 cKF_SkeletonInfo_R_init_standard_stop     c_keyframe.o
```

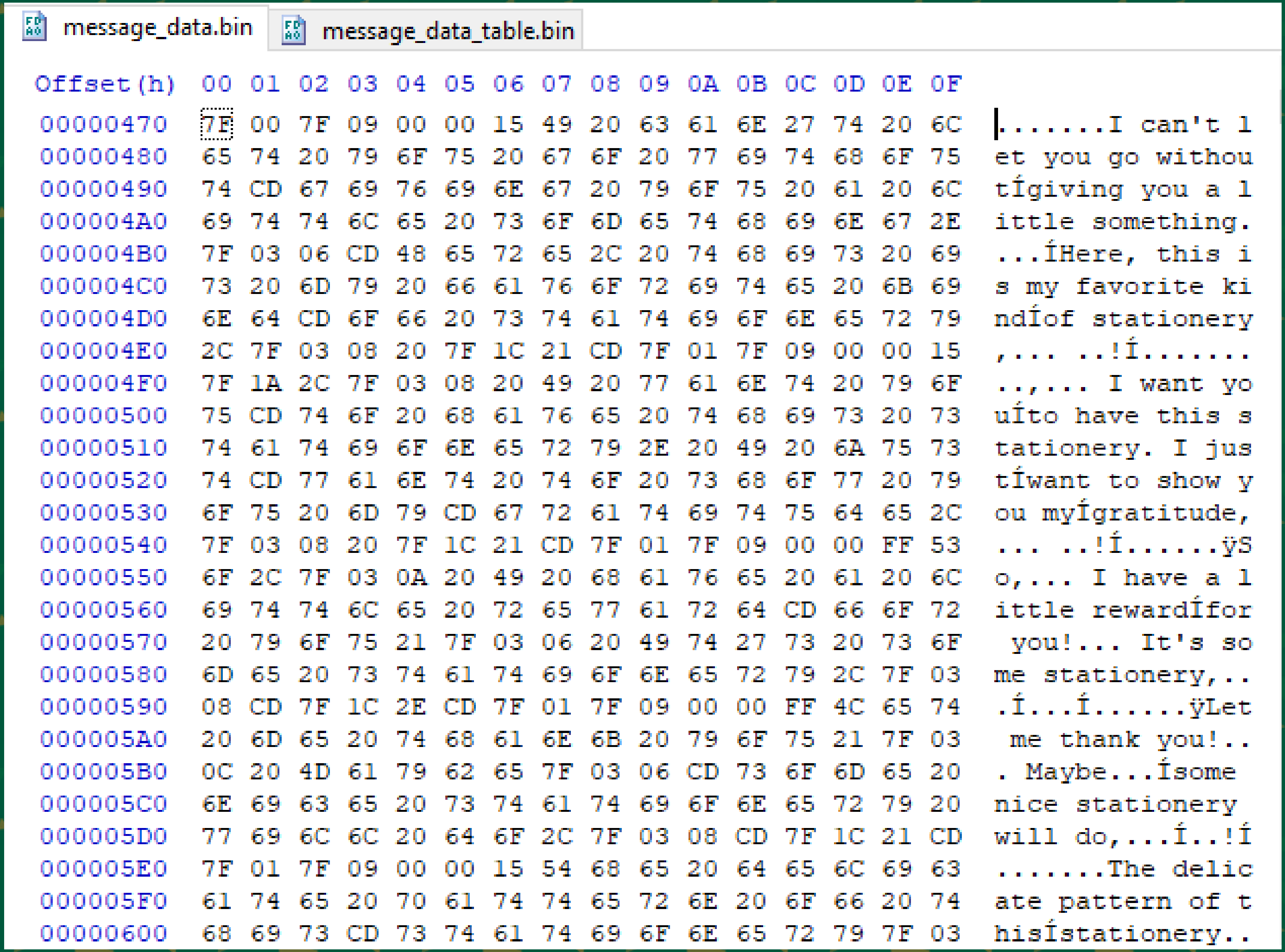
Reversing the Dialogue System

Reversing the Dialogue System

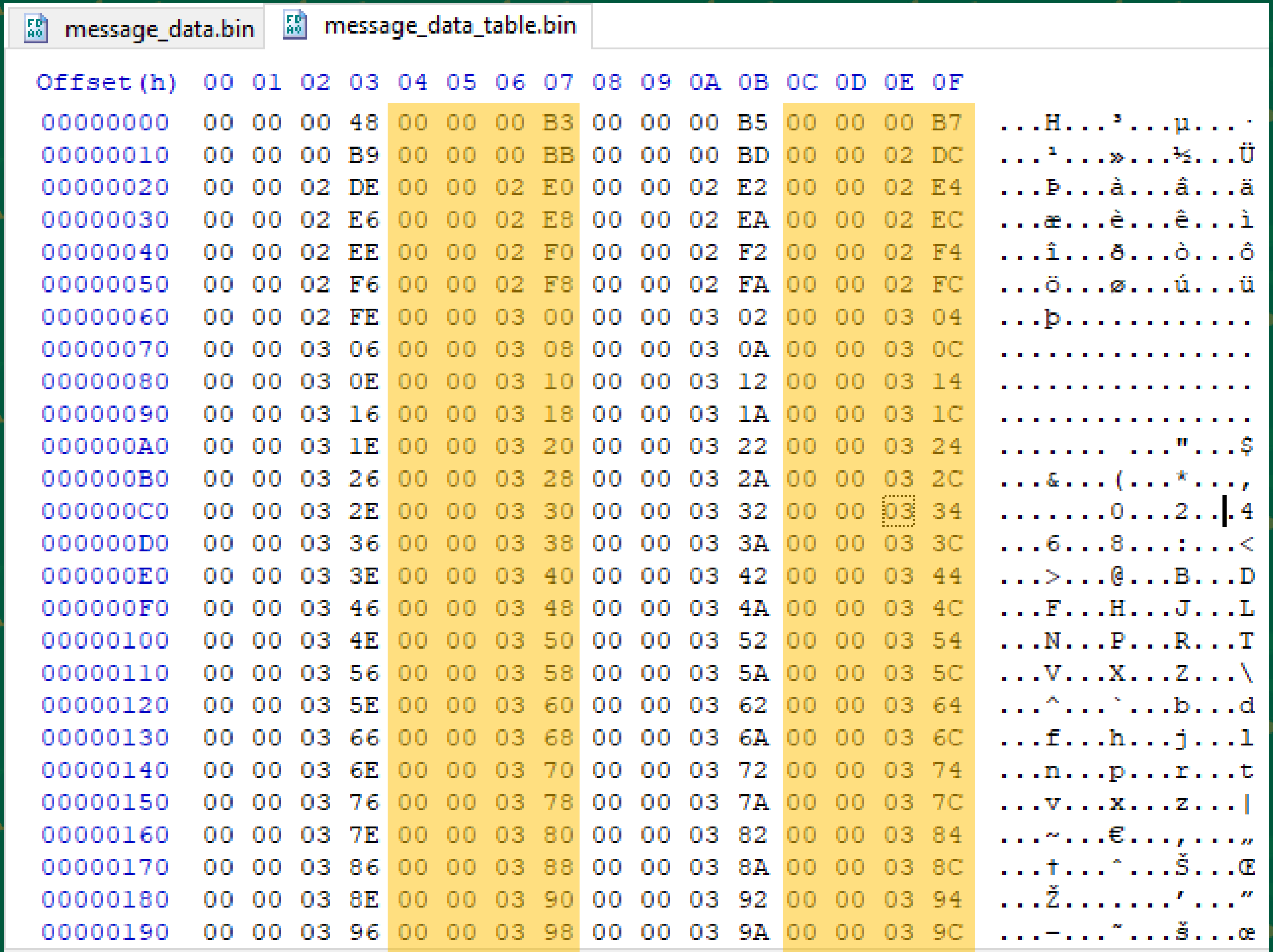
- Initial analysis
- Find the files that contain the message strings
  - `*_data.bin` and `*_data_table.bin` files



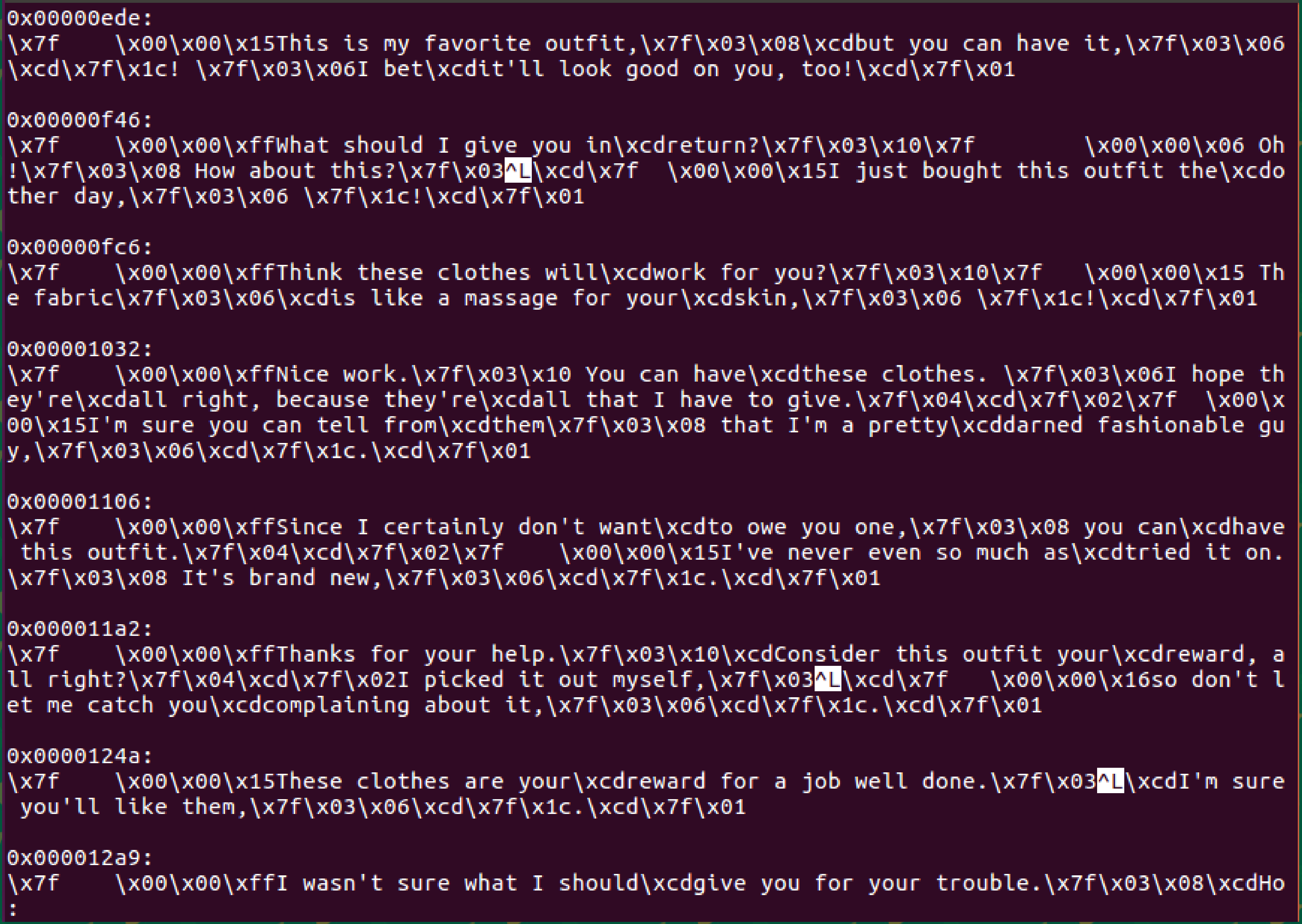
```
forest_1st.d/data/mail_data.bin
forest_1st.d/data/mail_data_table.bin
forest_1st.d/data/maila_data.bin
forest_1st.d/data/maila_data_table.bin
forest_1st.d/data/mailb_data.bin
forest_1st.d/data/mailb_data_table.bin
forest_1st.d/data/mailc_data.bin
forest_1st.d/data/mailc_data_table.bin
forest_1st.d/data/ps_data.bin
forest_1st.d/data/ps_data_table.bin
forest_1st.d/data/psz_data.bin
forest_1st.d/data/psz_data_table.bin
forest_1st.d/data/select_data.bin
forest_1st.d/data/select_data_table.bin
forest_1st.d/data/string_data.bin
forest_1st.d/data/string_data_table.bin
forest_1st.d/data/super_data.bin
forest_1st.d/data/super_data_table.bin
forest_1st.d/data/superz_data.bin
forest_1st.d/data/superz_data_table.bin
forest_2nd.d/data/message_data.bin
forest_2nd.d/data/message_data_table.bin
```



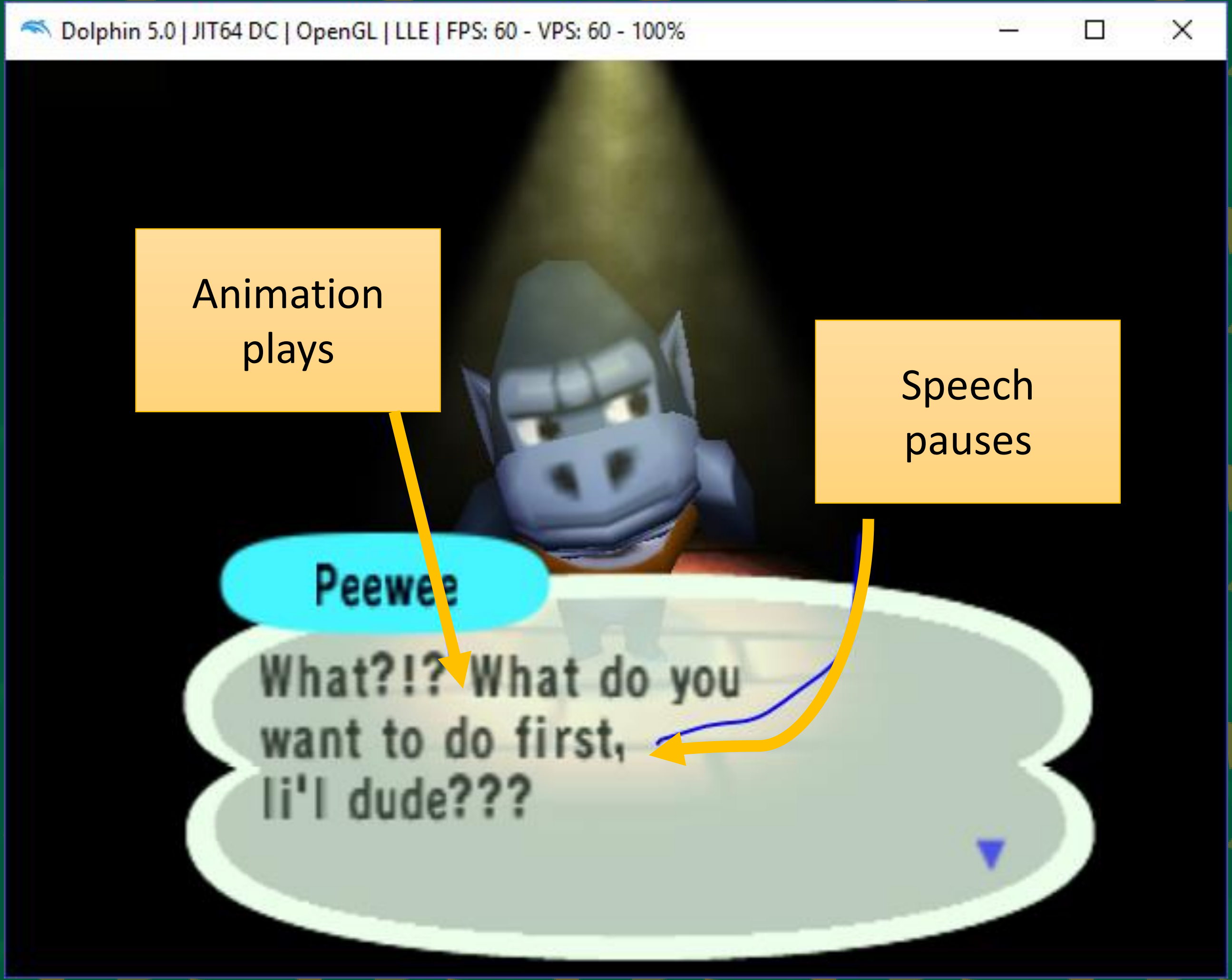
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000470  7F 00 7F 09 00 00 15 49 20 63 61 6E 27 74 20 6C  .......I can't l
00000480  65 74 20 79 6F 75 20 67 6F 20 77 69 74 68 6F 75  et you go withou
00000490  74 CD 67 69 76 69 6E 67 20 79 6F 75 20 61 20 6C  tÍgiving you a l
000004A0  69 74 74 6C 65 20 73 6F 6D 65 74 68 69 6E 67 2E  ittle something.
000004B0  7F 03 06 CD 48 65 72 65 2C 20 74 68 69 73 20 69  ...ÍHere, this i
000004C0  73 20 6D 79 20 66 61 76 6F 72 69 74 65 20 6B 69  s my favorite ki
000004D0  6E 64 CD 6F 66 20 73 74 61 74 69 6F 6E 65 72 79  ndÍof stationery
000004E0  2C 7F 03 08 20 7F 1C 21 CD 7F 01 7F 09 00 00 15  ,... ..!Í.......
000004F0  7F 1A 2C 7F 03 08 20 49 20 77 61 6E 74 20 79 6F  ..,... I want yo
00000500  75 CD 74 6F 20 68 61 76 65 20 74 68 69 73 20 73  uÍto have this s
00000510  74 61 74 69 6F 6E 65 72 79 2E 20 49 20 6A 75 73  tationery. I jus
00000520  74 CD 77 61 6E 74 20 74 6F 20 73 68 6F 77 20 79  tÍwant to show y
00000530  6F 75 20 6D 79 CD 67 72 61 74 69 74 75 64 65 2C  ou myÍgratitude,
00000540  7F 03 08 20 7F 1C 21 CD 7F 01 7F 09 00 00 FF 53  ... ..!Í......ÿS
00000550  6F 2C 7F 03 0A 20 49 20 68 61 76 65 20 61 20 6C  o,... I have a l
00000560  69 74 74 6C 65 20 72 65 77 61 72 64 CD 66 6F 72  ittle rewardÍfor
00000570  20 79 6F 75 21 7F 03 06 20 49 74 27 73 20 73 6F   you!... It's so
00000580  6D 65 20 73 74 61 74 69 6F 6E 65 72 79 2C 7F 03  me stationery,..
00000590  08 CD 7F 1C 2E CD 7F 01 7F 09 00 00 FF 4C 65 74  .Í...Í......ÿLet
000005A0  20 6D 65 20 74 68 61 6E 6B 20 79 6F 75 21 7F 03   me thank you!..
000005B0  0C 20 4D 61 79 62 65 7F 03 06 CD 73 6F 6D 65 20  . Maybe...Ísome
000005C0  6E 69 63 65 20 73 74 61 74 69 6F 6E 65 72 79 20  nice stationery
000005D0  77 69 6C 6C 20 64 6F 2C 7F 03 08 CD 7F 1C 21 CD  will do,...Í..!Í
000005E0  7F 01 7F 09 00 00 15 54 68 65 20 64 65 6C 69 63  .......The delic
000005F0  61 74 65 20 70 61 74 74 65 72 6E 20 6F 66 20 74  ate pattern of t
00000600  68 69 73 CD 73 74 61 74 69 6F 6E 65 72 79 7F 03  hisÍstationery..
```



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  00 00 00 48 00 00 00 B3 00 00 00 B5 00 00 00 B7  ...H...³...µ...·
00000010  00 00 00 B9 00 00 00 BB 00 00 00 BD 00 00 02 DC  ...¹...»...½...Ü
00000020  00 00 02 DE 00 00 02 E0 00 00 02 E2 00 00 02 E4  ...Þ...à...â...ä
00000030  00 00 02 E6 00 00 02 E8 00 00 02 EA 00 00 02 EC  ...æ...è...ê...ì
00000040  00 00 02 EE 00 00 02 F0 00 00 02 F2 00 00 02 F4  ...î...ð...ò...ô
00000050  00 00 02 F6 00 00 02 F8 00 00 02 FA 00 00 02 FC  ...ö...ø...ú...ü
00000060  00 00 02 FE 00 00 03 00 00 00 03 02 00 00 03 04  ...þ............
00000070  00 00 03 06 00 00 03 08 00 00 03 0A 00 00 03 0C  ................
00000080  00 00 03 0E 00 00 03 10 00 00 03 12 00 00 03 14  ................
00000090  00 00 03 16 00 00 03 18 00 00 03 1A 00 00 03 1C  ................
000000A0  00 00 03 1E 00 00 03 20 00 00 03 22 00 00 03 24  ....... ..."...$
000000B0  00 00 03 26 00 00 03 28 00 00 03 2A 00 00 03 2C  ...&...(...*...,
000000C0  00 00 03 2E 00 00 03 30 00 00 03 32 00 00 03 34  .......0...2...4
000000D0  00 00 03 36 00 00 03 38 00 00 03 3A 00 00 03 3C  ...6...8...:...<
000000E0  00 00 03 3E 00 00 03 40 00 00 03 42 00 00 03 44  ...>...@...B...D
000000F0  00 00 03 46 00 00 03 48 00 00 03 4A 00 00 03 4C  ...F...H...J...L
00000100  00 00 03 4E 00 00 03 50 00 00 03 52 00 00 03 54  ...N...P...R...T
00000110  00 00 03 56 00 00 03 58 00 00 03 5A 00 00 03 5C  ...V...X...Z...\
00000120  00 00 03 5E 00 00 03 60 00 00 03 62 00 00 03 64  ...^...`...b...d
00000130  00 00 03 66 00 00 03 68 00 00 03 6A 00 00 03 6C  ...f...h...j...l
00000140  00 00 03 6E 00 00 03 70 00 00 03 72 00 00 03 74  ...n...p...r...t
00000150  00 00 03 76 00 00 03 78 00 00 03 7A 00 00 03 7C  ...v...x...z...|
00000160  00 00 03 7E 00 00 03 80 00 00 03 82 00 00 03 84  ...~...€...‚...„
00000170  00 00 03 86 00 00 03 88 00 00 03 8A 00 00 03 8C  ...†...ˆ...Š...Œ
00000180  00 00 03 8E 00 00 03 90 00 00 03 92 00 00 03 94  ...Ž.......’...”
00000190  00 00 03 96 00 00 03 98 00 00 03 9A 00 00 03 9C  ...–...˜...š...œ
```

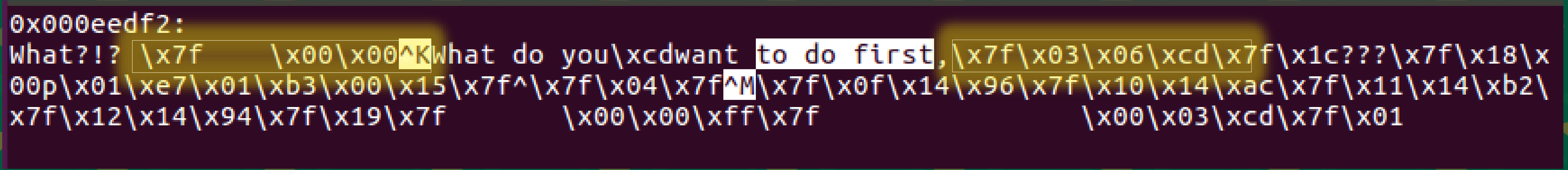


```
0x00000ede:
\x7f    \x00\x00\x15This is my favorite outfit,\x7f\x03\x08\xcdbut you can have it,\x7f\x03\x06
\xcd\x7f\x1c! \x7f\x03\x06I bet\xcdit'll look good on you, too!\xcd\x7f\x01

0x00000f46:
\x7f    \x00\x00\xffWhat should I give you in\xcdreturn?\x7f\x03\x10\x7f        \x00\x00\x06 Oh
!\x7f\x03\x08 How about this?\x7f\x03^L\xcd\x7f  \x00\x00\x15I just bought this outfit the\xcdo
ther day,\x7f\x03\x06 \x7f\x1c!\xcd\x7f\x01

0x00000fc6:
\x7f    \x00\x00\xffThink these clothes will\xcdwork for you?\x7f\x03\x10\x7f   \x00\x00\x15 Th
e fabric\x7f\x03\x06\xcdis like a massage for your\xcdskin,\x7f\x03\x06 \x7f\x1c!\xcd\x7f\x01

0x00001032:
\x7f    \x00\x00\xffNice work.\x7f\x03\x10 You can have\xcdthese clothes. \x7f\x03\x06I hope th
ey're\xcdall right, because they're\xcdall that I have to give.\x7f\x04\xcd\x7f\x02\x7f  \x00\x
00\x15I'm sure you can tell from\xcdthem\x7f\x03\x08 that I'm a pretty\xcddarned fashionable gu
y,\x7f\x03\x06\xcd\x7f\x1c.\xcd\x7f\x01

0x00001106:
\x7f    \x00\x00\xffSince I certainly don't want\xcdto owe you one,\x7f\x03\x08 you can\xcdhave
 this outfit.\x7f\x04\xcd\x7f\x02\x7f    \x00\x00\x15I've never even so much as\xcdtried it on.
\x7f\x03\x08 It's brand new,\x7f\x03\x06\xcd\x7f\x1c.\xcd\x7f\x01

0x000011a2:
\x7f    \x00\x00\xffThanks for your help.\x7f\x03\x10\xcdConsider this outfit your\xcdreward, a
ll right?\x7f\x04\xcd\x7f\x02I picked it out myself,\x7f\x03^L\xcd\x7f   \x00\x00\x16so don't l
et me catch you\xcdcomplaining about it,\x7f\x03\x06\xcd\x7f\x1c.\xcd\x7f\x01

0x0000124a:
\x7f    \x00\x00\x15These clothes are your\xcdreward for a job well done.\x7f\x03^L\xcdI'm sure
 you'll like them,\x7f\x03\x06\xcd\x7f\x1c.\xcd\x7f\x01

0x000012a9:
\x7f    \x00\x00\xffI wasn't sure what I should\xcdgive you for your trouble.\x7f\x03\x08\xcdHo
:
```

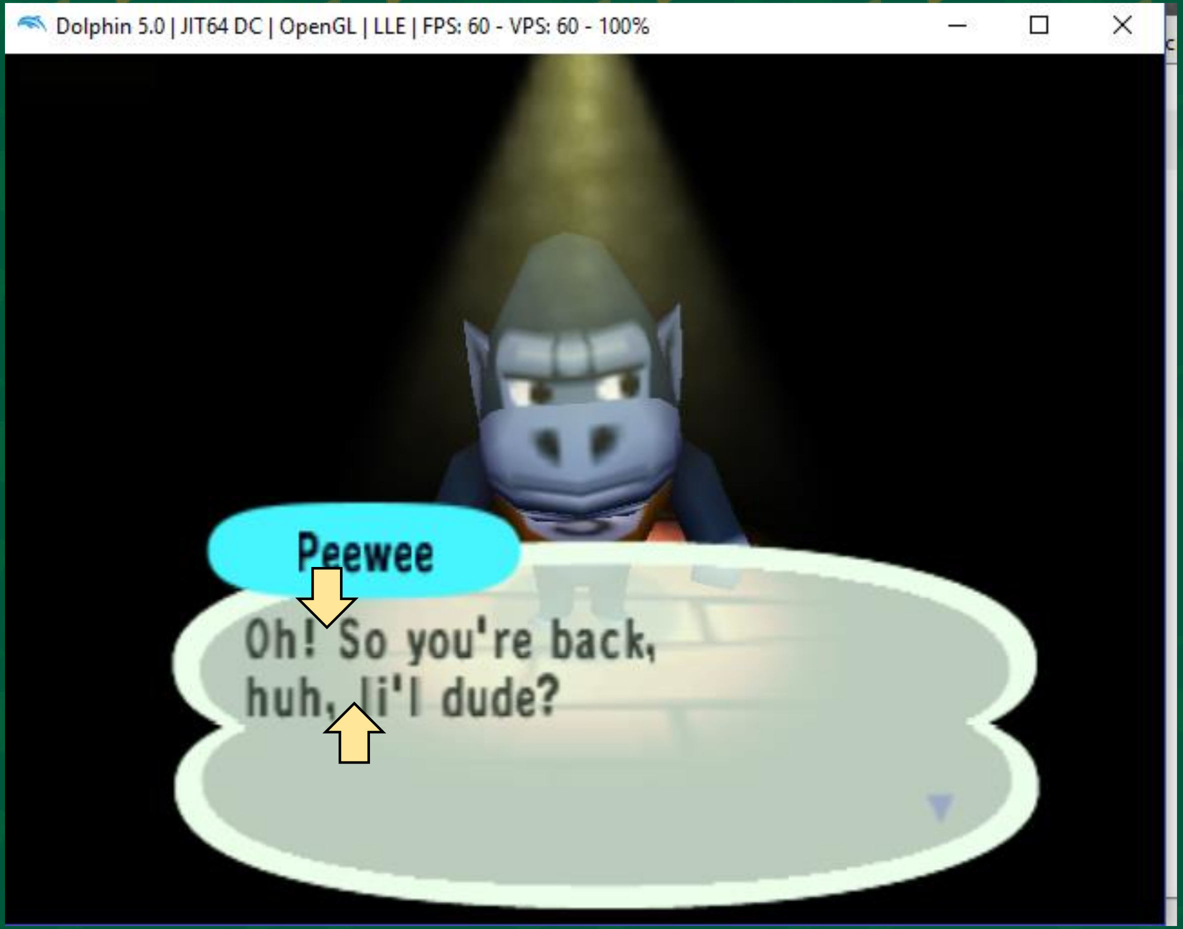

Dolphin 5.0 | JIT64 DC | OpenGL | LLE | FPS: 60 - VPS: 60 - 100%
Peewee
What?!? What do you want to do first, li'l dude???

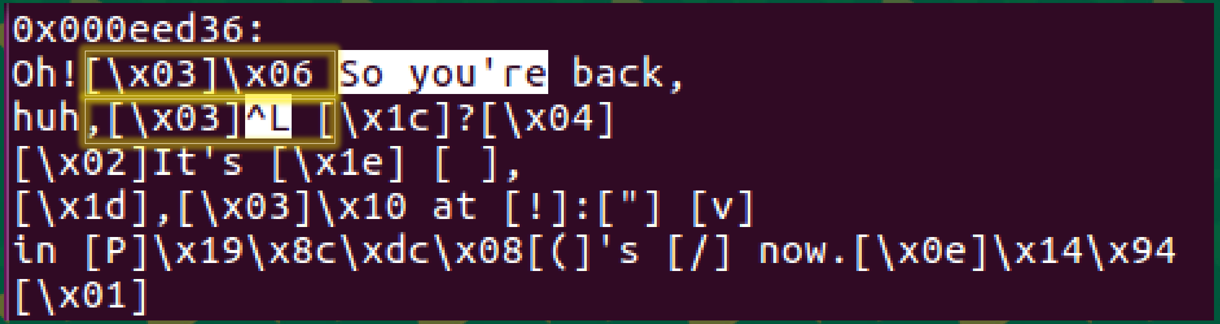
Animation plays

Speech pauses

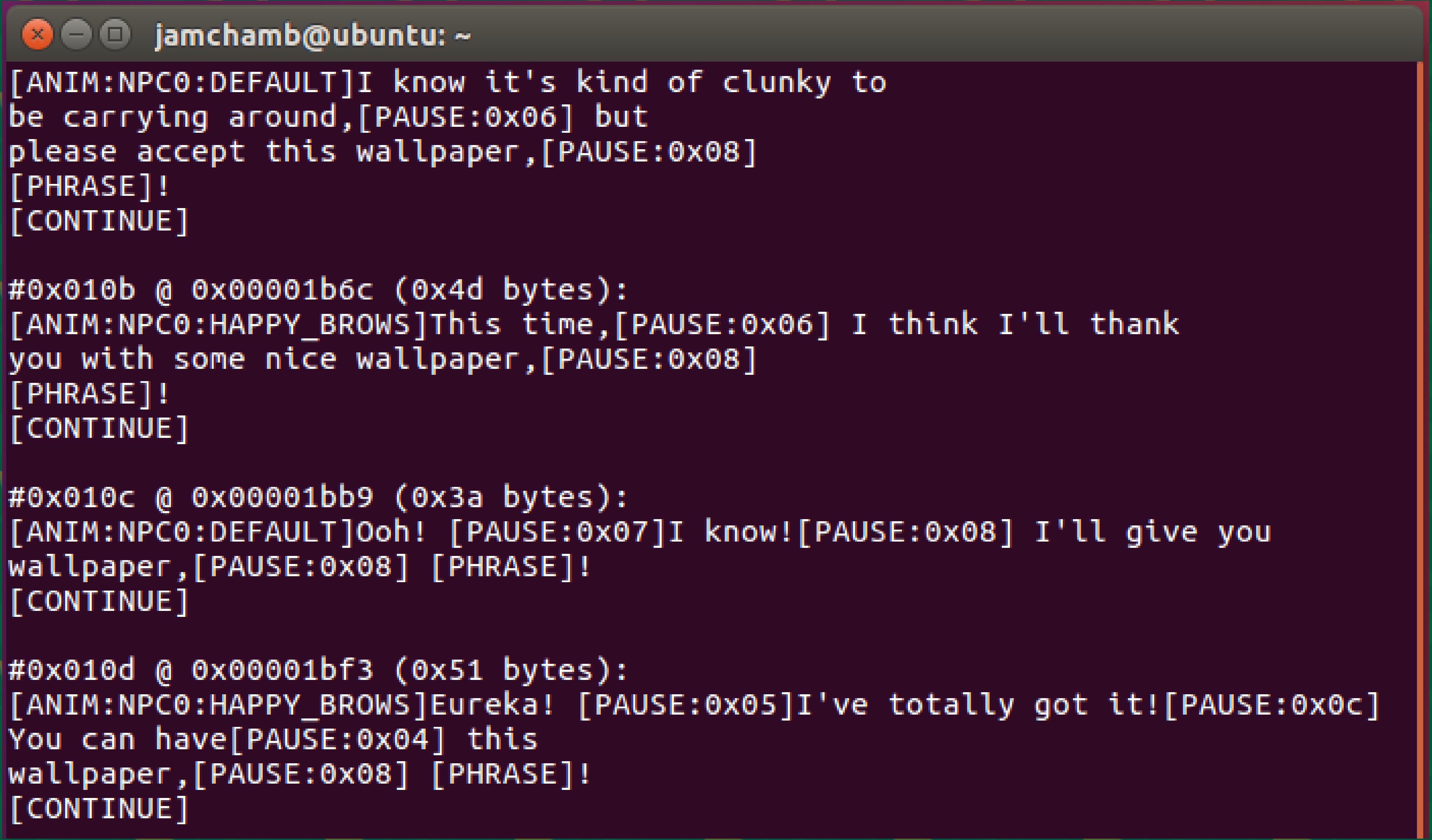


```
0x000eedf2:
What?!? \x7f    \x00\x00^KWhat do you\xcdwant to do first,\x7f\x03\x06\xcd\x7f\x1c???\x7f\x18\x
00p\x01\xe7\x01\xb3\x00\x15\x7f^\x7f\x04\x7f^M\x7f\x0f\x14\x96\x7f\x10\x14\xac\x7f\x11\x14\xb2\
x7f\x12\x14\x94\x7f\x19\x7f       \x00\x00\xff\x7f                \x00\x03\xcd\x7f\x01
```


Dolphin 5.0 | JIT64 DC | OpenGL | LLE | FPS: 60 - VPS: 60 - 100%
Peewee
Oh! So you're back,
huh, li'l dude?



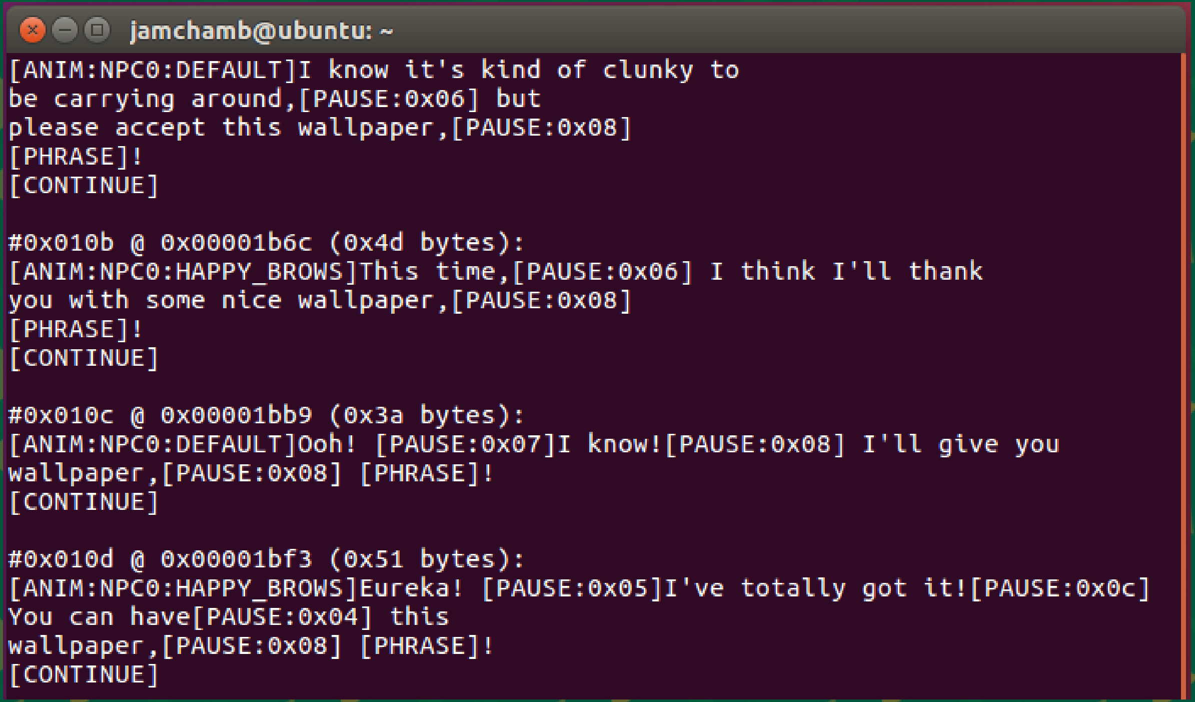
```
0x000eed36:
Oh![\x03]\x06 So you're back,
huh,[\x03]^L [\x1c]?[\x04]
[\x02]It's [\x1e] [ ],
[\x1d],[\x03]\x10 at [!]:["] [v]
in [P]\x19\x8c\xdc\x08[(]'s [/] now.[\x0e]\x14\x94
[\x01]
```



```
[ANIM:NPC0:DEFAULT]I know it's kind of clunky to
be carrying around,[PAUSE:0x06] but
please accept this wallpaper,[PAUSE:0x08]
[PHRASE]!
[CONTINUE]

#0x010b @ 0x00001b6c (0x4d bytes):
[ANIM:NPC0:HAPPY_BROWS]This time,[PAUSE:0x06] I think I'll thank
you with some nice wallpaper,[PAUSE:0x08]
[PHRASE]!
[CONTINUE]

#0x010c @ 0x00001bb9 (0x3a bytes):
[ANIM:NPC0:DEFAULT]Ooh! [PAUSE:0x07]I know![PAUSE:0x08] I'll give you
wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]

#0x010d @ 0x00001bf3 (0x51 bytes):
[ANIM:NPC0:HAPPY_BROWS]Eureka! [PAUSE:0x05]I've totally got it![PAUSE:0x0c]
You can have[PAUSE:0x04] this
wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]
```

Reversing the Dialogue System

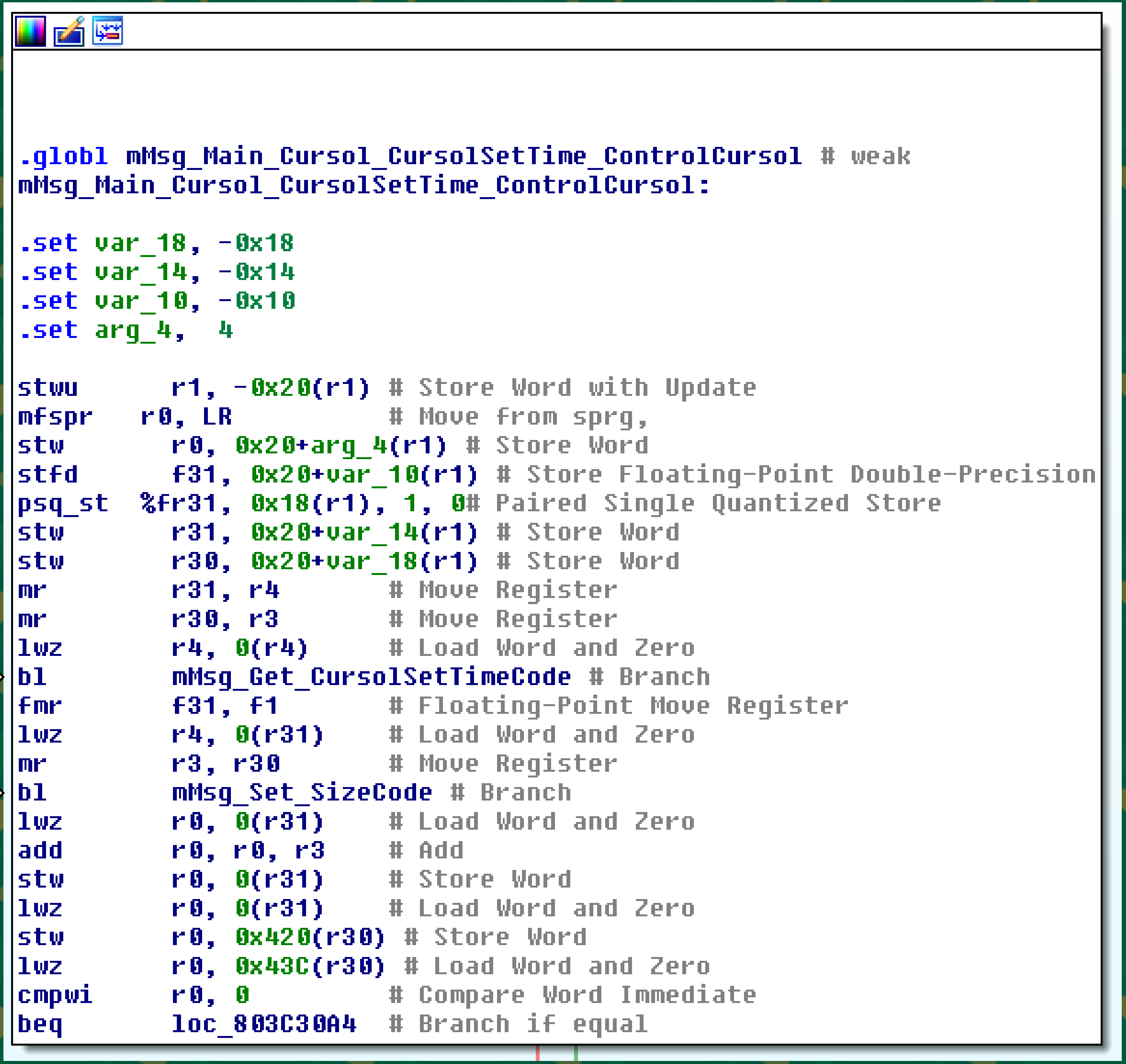
- Iteratively add codes
  - Focus on what non-printable bytes are left
- Basic editor
  - Doesn’t have all the codes defined
  - Doesn’t support writing special codes back
  - Good for analysis
- Use IDA to figure out the rest of the special codes



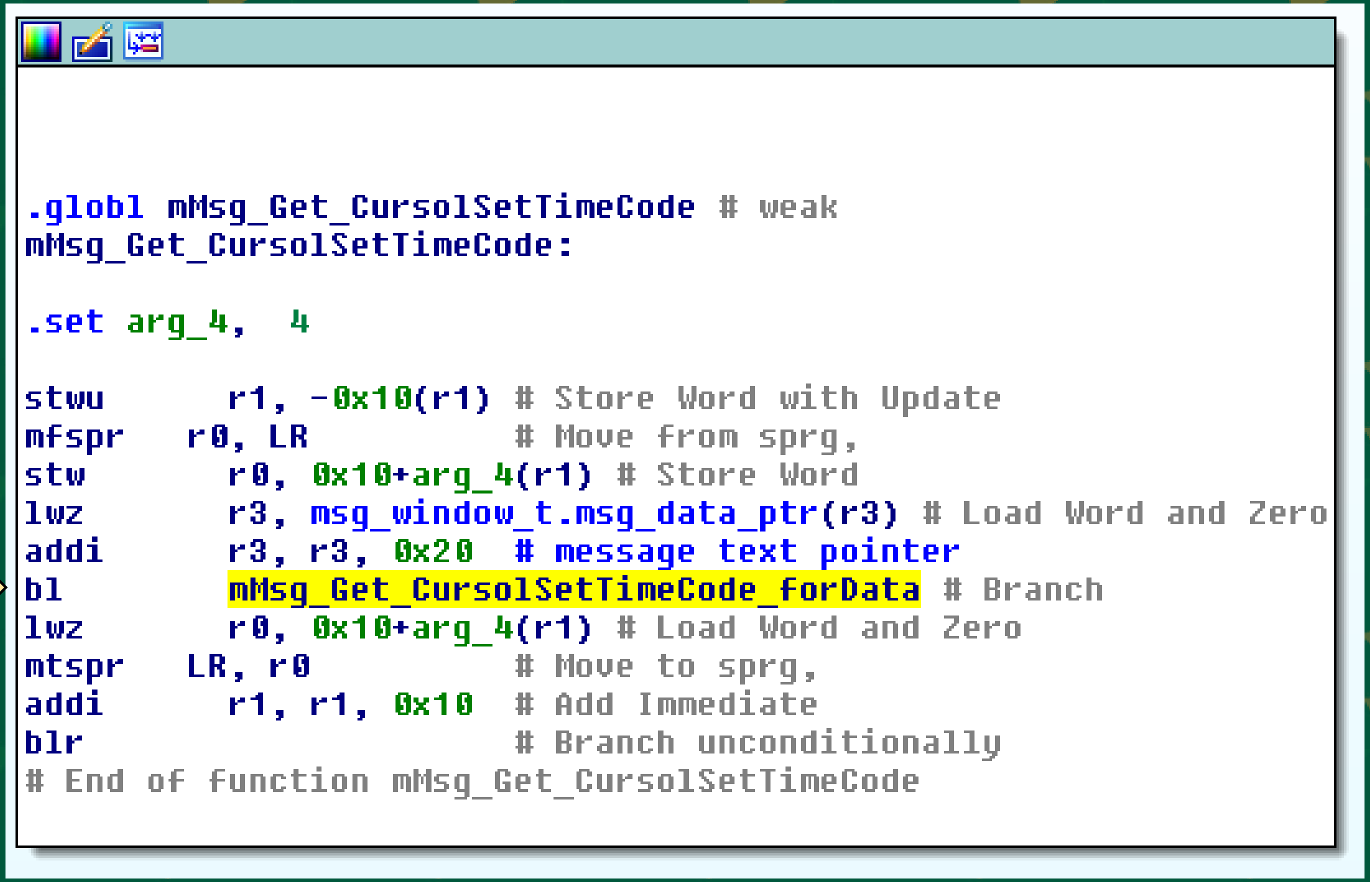
```
[ANIM:NPC0:DEFAULT]I know it's kind of clunky to
be carrying around,[PAUSE:0x06] but
please accept this wallpaper,[PAUSE:0x08]
[PHRASE]!
[CONTINUE]

#0x010b @ 0x00001b6c (0x4d bytes):
[ANIM:NPC0:HAPPY_BROWS]This time,[PAUSE:0x06] I think I'll thank
you with some nice wallpaper,[PAUSE:0x08]
[PHRASE]!
[CONTINUE]

#0x010c @ 0x00001bb9 (0x3a bytes):
[ANIM:NPC0:DEFAULT]Ooh! [PAUSE:0x07]I know![PAUSE:0x08] I'll give you
wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]

#0x010d @ 0x00001bf3 (0x51 bytes):
[ANIM:NPC0:HAPPY_BROWS]Eureka! [PAUSE:0x05]I've totally got it![PAUSE:0x0c]
You can have[PAUSE:0x04] this
wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]
```

Reversing the Dialogue System

- “ControlCursol” functions for each code handle reading the special bytes and doing something with them

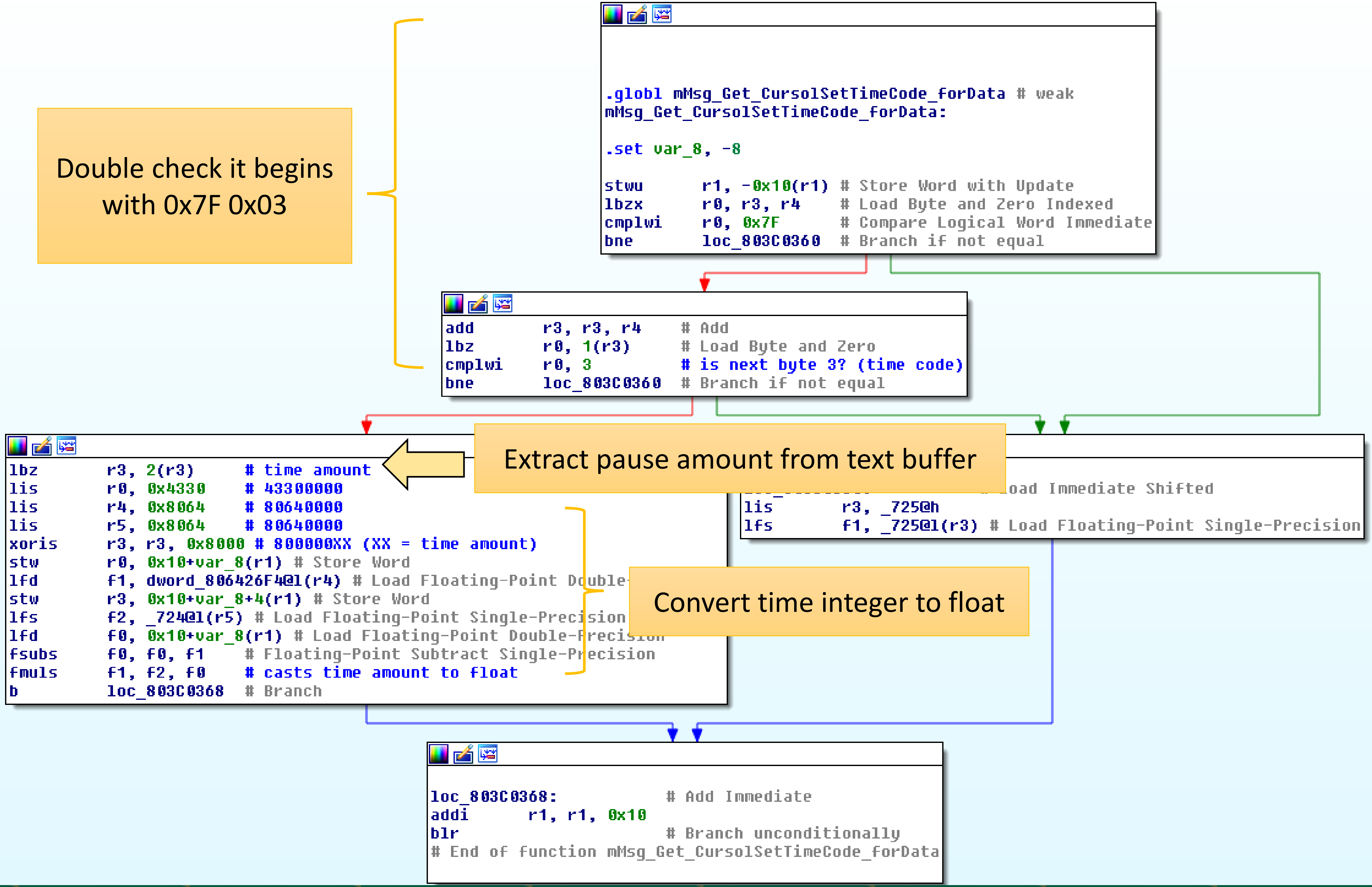


```
.globl mMsg_Main_Cursol_CursolSetTime_ControlCursol # weak
mMsg_Main_Cursol_CursolSetTime_ControlCursol:

.set var_18, -0x18
.set var_14, -0x14
.set var_10, -0x10
.set arg_4,  4

stwu     r1, -0x20(r1) # Store Word with Update
mfspr   r0, LR         # Move from sprg,
stw      r0, 0x20+arg_4(r1) # Store Word
stfd     f31, 0x20+var_10(r1) # Store Floating-Point Double-Precision
psq_st  %fr31, 0x18(r1), 1, 0# Paired Single Quantized Store
stw      r31, 0x20+var_14(r1) # Store Word
stw      r30, 0x20+var_18(r1) # Store Word
mr       r31, r4       # Move Register
mr       r30, r3       # Move Register
lwz      r4, 0(r4)     # Load Word and Zero
bl       mMsg_Get_CursolSetTimeCode # Branch
fmr      f31, f1       # Floating-Point Move Register
lwz      r4, 0(r31)    # Load Word and Zero
mr       r3, r30       # Move Register
bl       mMsg_Set_SizeCode # Branch
lwz      r0, 0(r31)    # Load Word and Zero
add      r0, r0, r3    # Add
stw      r0, 0(r31)    # Store Word
lwz      r0, 0(r31)    # Load Word and Zero
stw      r0, 0x420(r30) # Store Word
lwz      r0, 0x43C(r30) # Load Word and Zero
cmpwi    r0, 0         # Compare Word Immediate
beq      loc_803C30A4  # Branch if equal
```



```
.globl mMsg_Get_CursolSetTimeCode # weak
mMsg_Get_CursolSetTimeCode:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mfspr   r0, LR          # Move from sprg,
stw       r0, 0x10+arg_4(r1) # Store Word
lwz       r3, msg_window_t.msg_data_ptr(r3) # Load Word and Zero
addi      r3, r3, 0x20  # message text pointer
bl        mMsg_Get_CursolSetTimeCode_forData # Branch
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
mtspr   LR, r0          # Move to sprg,
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
# End of function mMsg_Get_CursolSetTimeCode
```

Double check it begins with 0x7F 0x03



```
.globl mMsg_Get_CursolSetTimeCode_forData # weak
mMsg_Get_CursolSetTimeCode_forData:

.set var_8, -8

stwu      r1, -0x10(r1) # Store Word with Update
lbzx      r0, r3, r4    # Load Byte and Zero Indexed
cmplwi    r0, 0x7F      # Compare Logical Word Immediate
bne       loc_803C0360  # Branch if not equal
```

```
add       r3, r3, r4    # Add
lbz       r0, 1(r3)     # Load Byte and Zero
cmplwi    r0, 3         # is next byte 3? (time code)
bne       loc_803C0360  # Branch if not equal
```

```
lbz       r3, 2(r3)     # time amount
lis       r0, 0x4330    # 43300000
lis       r4, 0x8064    # 80640000
lis       r5, 0x8064    # 80640000
xoris     r3, r3, 0x8000 # 800000XX (XX = time amount)
stw       r0, 0x10+var_8(r1) # Store Word
lfd       f1, dword_806426F4@l(r4) # Load Floating-Point Double-
stw       r3, 0x10+var_8+4(r1) # Store Word
lfs       f2, _724@l(r5) # Load Floating-Point Single-Precision
lfd       f0, 0x10+var_8(r1) # Load Floating-Point Double-Precision
fsubs     f0, f0, f1    # Floating-Point Subtract Single-Precision
fmuls     f1, f2, f0    # casts time amount to float
b         loc_803C0368  # Branch
```

```
                        # Load Immediate Shifted
lis       r3, _725@h
lfs       f1, _725@l(r3) # Load Floating-Point Single-Precision
```

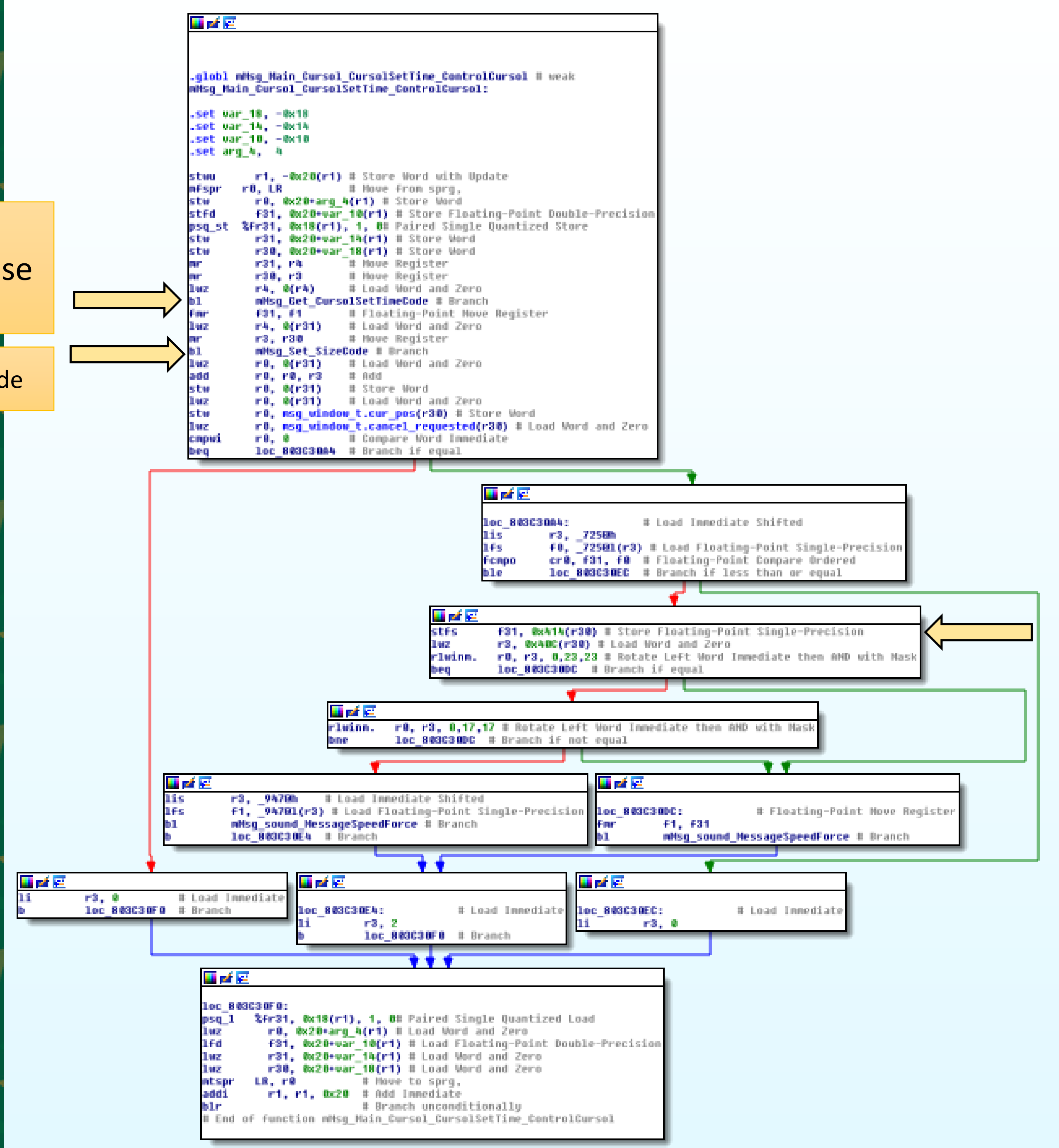
```
loc_803C0368:           # Add Immediate
addi      r1, r1, 0x10
blr                     # Branch unconditionally
# End of function mMsg_Get_CursolSetTimeCode_forData
```

Extract pause amount from text buffer

Convert time integer to float

Return from extracting pause interval

Report size of the code



```
.globl mMsg_Main_Cursol_CursolSetTime_ControlCursol # weak
mMsg_Main_Cursol_CursolSetTime_ControlCursol:

.set var_18, -0x18
.set var_14, -0x14
.set var_10, -0x10
.set arg_4,  4

stwu      r1, -0x20(r1) # Store Word with Update
mfspr   r0, LR          # Move From sprg,
stw       r0, 0x20+arg_4(r1) # Store Word
stfd      f31, 0x20+var_10(r1) # Store Floating-Point Double-Precision
psq_st  %fr31, 0x18(r1), 1, 0# Paired Single Quantized Store
stw       r31, 0x20+var_14(r1) # Store Word
stw       r30, 0x20+var_18(r1) # Store Word
mr        r31, r4         # Move Register
mr        r30, r3         # Move Register
lwz       r4, 0(r4)       # Load Word and Zero
bl        mMsg_Get_CursolSetTimeCode # Branch
fmr       f31, f1         # Floating-Point Move Register
lwz       r4, 0(r31)      # Load Word and Zero
mr        r3, r30         # Move Register
bl        mMsg_Set_SizeCode # Branch
lwz       r0, 0(r31)      # Load Word and Zero
add       r0, r0, r3      # Add
stw       r0, 0(r31)      # Store Word
lwz       r0, 0(r31)      # Load Word and Zero
stw       r0, msg_window_t.cur_pos(r30) # Store Word
lwz       r0, msg_window_t.cancel_requested(r30) # Load Word and Zero
cmpwi     r0, 0           # Compare Word Immediate
beq       loc_803C30A4    # Branch if equal
```

```
loc_803C30A4:              # Load Immediate Shifted
lis        r3, _7250h
lfs        f0, _72501(r3) # Load Floating-Point Single-Precision
fcmpo      cr0, f31, f0  # Floating-Point Compare Ordered
ble        loc_803C30EC  # Branch if less than or equal
```

```
stfs       f31, 0x414(r30) # Store Floating-Point Single-Precision
lwz        r3, 0x40C(r30) # Load Word and Zero
rlwinm.    r0, r3, 0,23,23 # Rotate Left Word Immediate then AND with Mask
beq        loc_803C30DC  # Branch if equal
```

```
rlwinm.    r0, r3, 0,17,17 # Rotate Left Word Immediate then AND with Mask
bne        loc_803C30DC  # Branch if not equal
```

```
lis        r3, _9470h     # Load Immediate Shifted
lfs        f1, _94701(r3) # Load Floating-Point Single-Precision
bl         mMsg_sound_MessageSpeedForce # Branch
b          loc_803C30E4  # Branch
```

```
loc_803C30DC:              # Floating-Point Move Register
fmr        f1, f31
bl         mMsg_sound_MessageSpeedForce # Branch
```

```
li         r3, 0          # Load Immediate
b          loc_803C30F0  # Branch
```

```
loc_803C30E4:              # Load Immediate
li         r3, 2
b          loc_803C30F0  # Branch
```

```
loc_803C30EC:              # Load Immediate
li         r3, 0
```

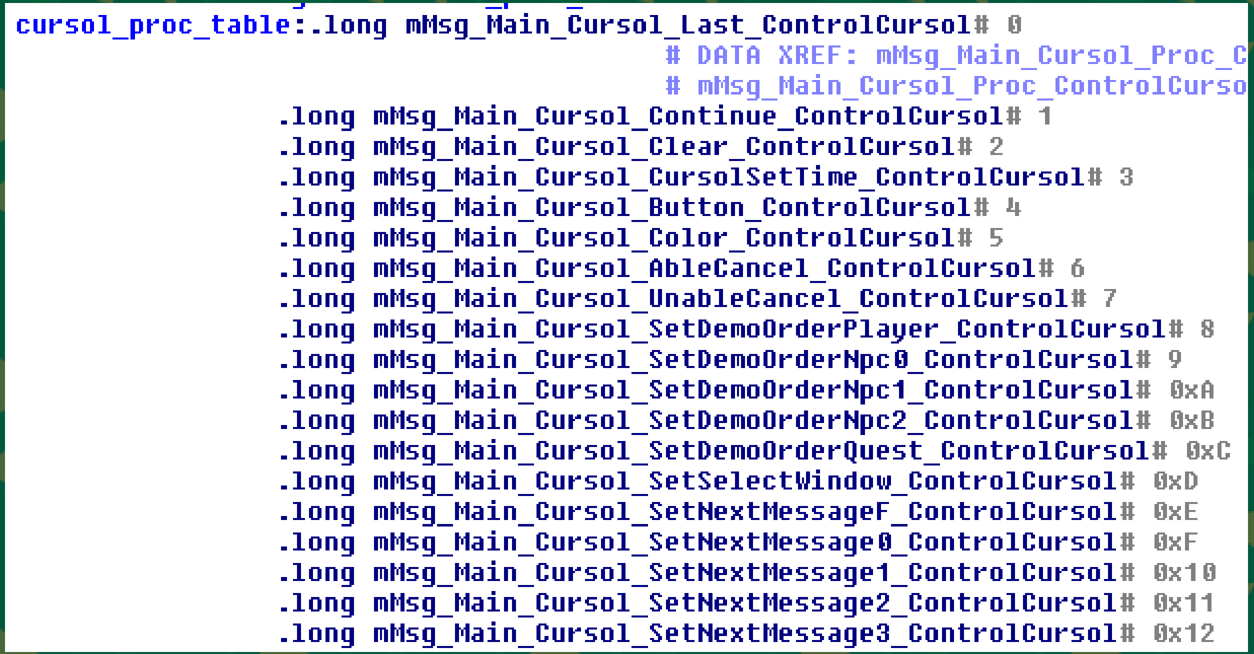
```
loc_803C30F0:
psq_l   %fr31, 0x18(r1), 1, 0# Paired Single Quantized Load
lwz       r0, 0x20+arg_4(r1) # Load Word and Zero
lfd       f31, 0x20+var_10(r1) # Load Floating-Point Double-Precision
lwz       r31, 0x20+var_14(r1) # Load Word and Zero
lwz       r30, 0x20+var_18(r1) # Load Word and Zero
mtspr   LR, r0          # Move to sprg,
addi      r1, r1, 0x20  # Add Immediate
blr                     # Branch unconditionally
# End of function mMsg_Main_Cursol_CursolSetTime_ControlCursol
```

Save interval to `msg_window_t.timer`

Do stuff with extracted data for code

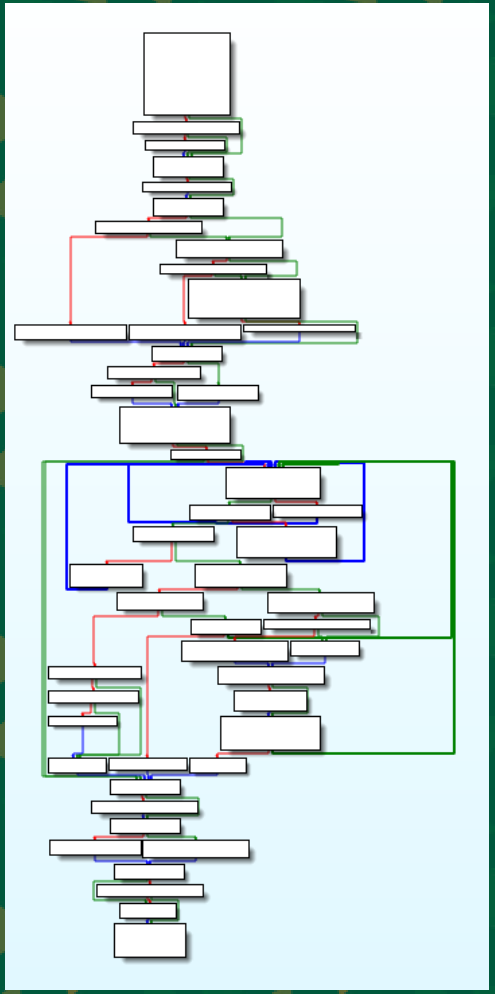
Reversing the Dialogue System

- Still don't know where these cursor controller functions are used
    - Cross-references are a dead end
- Search for the function addresses as raw bytes…
    - They're held in a function table
    - Data type of the bytes wasn't defined, so the references didn't show up
- Referenced by `mMsg_Main_Cursol_Proc_ControlCursol`
    - Performs the table lookup by code
    - Referenced in turn by `mMsg_Main_Cursol_ControlCursol`



```
cursol_proc_table:.long mMsg_Main_Cursol_Last_ControlCursol# 0
                                        # DATA XREF: mMsg_Main_Cursol_Proc_C
                                        # mMsg_Main_Cursol_Proc_ControlCurso
                .long mMsg_Main_Cursol_Continue_ControlCursol# 1
                .long mMsg_Main_Cursol_Clear_ControlCursol# 2
                .long mMsg_Main_Cursol_CursolSetTime_ControlCursol# 3
                .long mMsg_Main_Cursol_Button_ControlCursol# 4
                .long mMsg_Main_Cursol_Color_ControlCursol# 5
                .long mMsg_Main_Cursol_AbleCancel_ControlCursol# 6
                .long mMsg_Main_Cursol_UnableCancel_ControlCursol# 7
                .long mMsg_Main_Cursol_SetDemoOrderPlayer_ControlCursol# 8
                .long mMsg_Main_Cursol_SetDemoOrderNpc0_ControlCursol# 9
                .long mMsg_Main_Cursol_SetDemoOrderNpc1_ControlCursol# 0xA
                .long mMsg_Main_Cursol_SetDemoOrderNpc2_ControlCursol# 0xB
                .long mMsg_Main_Cursol_SetDemoOrderQuest_ControlCursol# 0xC
                .long mMsg_Main_Cursol_SetSelectWindow_ControlCursol# 0xD
                .long mMsg_Main_Cursol_SetNextMessageF_ControlCursol# 0xE
                .long mMsg_Main_Cursol_SetNextMessage0_ControlCursol# 0xF
                .long mMsg_Main_Cursol_SetNextMessage1_ControlCursol# 0x10
                .long mMsg_Main_Cursol_SetNextMessage2_ControlCursol# 0x11
                .long mMsg_Main_Cursol_SetNextMessage3_ControlCursol# 0x12
```

mMsg_Main_Cursol_ControlCursol



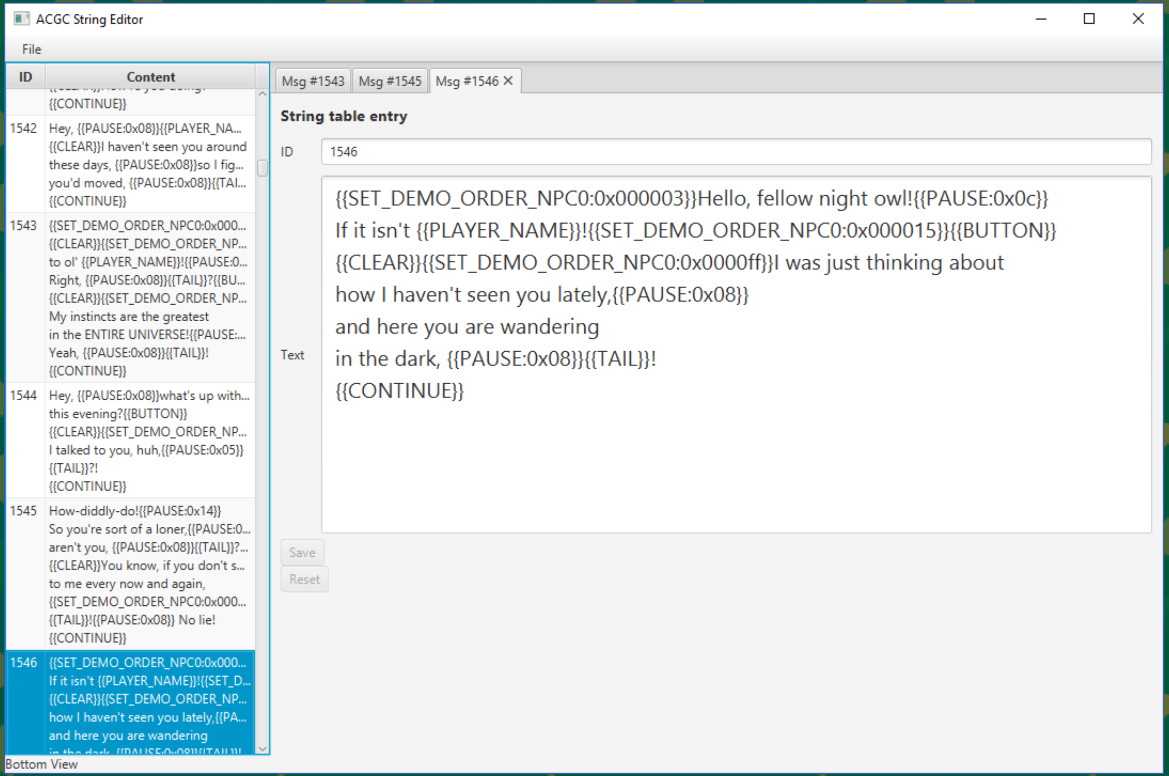
Timing and cancel request handling

Check data at cursor;
Print or handle proc code

Timing and talk animation/sound handling

Reversing the Dialogue System

- GUI editor for string tables
- Translates special codes to serialized text format, e.g. {{PAUSE:0x03}}
- Handles special character set
- Note: Adding more entries to a table requires generating a patch
    - Highest entry ID is compiled in, and used for bounds check



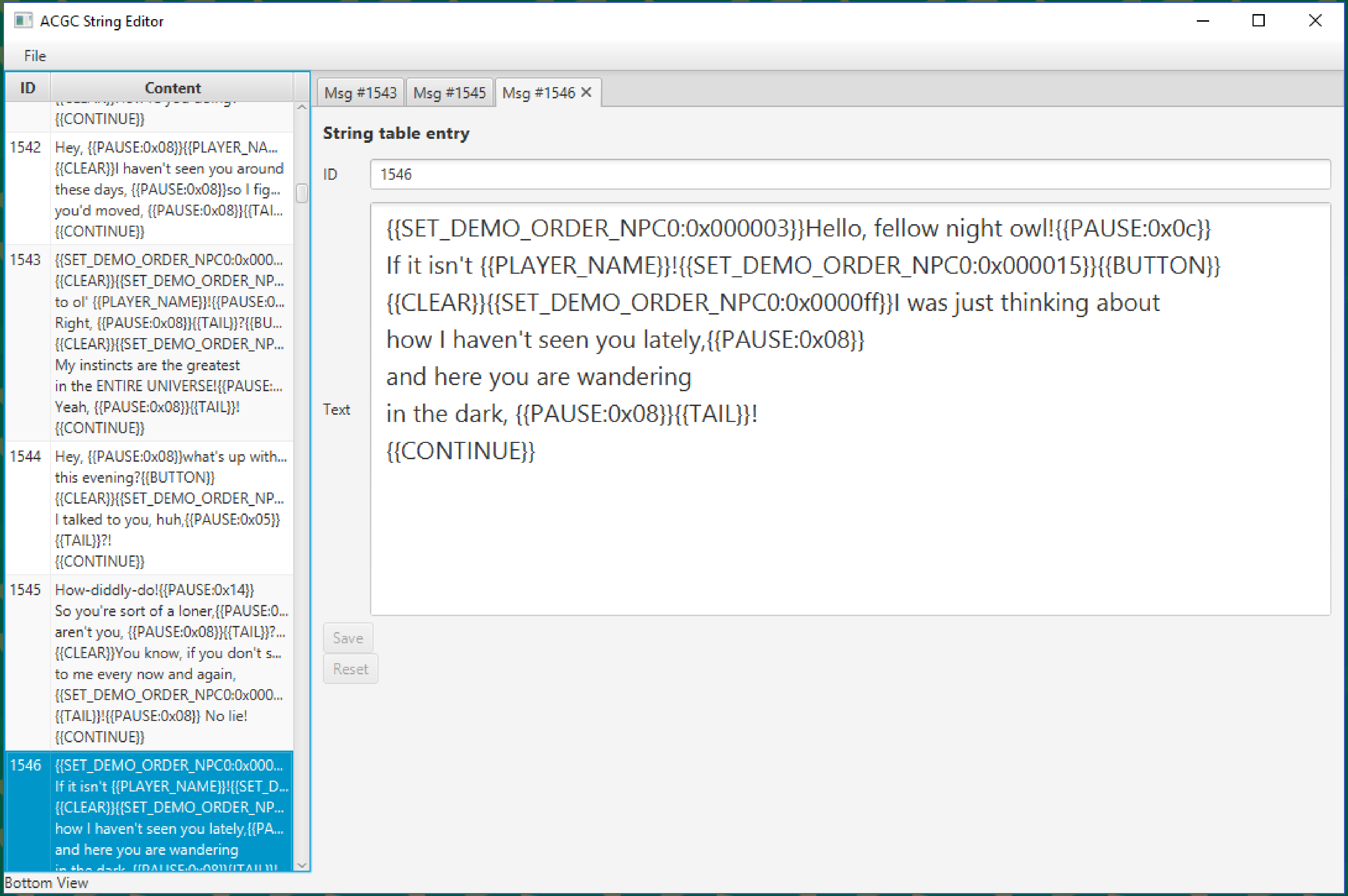
ID	Content
	{{CONTINUE}}
1542	Hey, {{PAUSE:0x08}}{{PLAYER_NA... {{CLEAR}}I haven't seen you around these days, {{PAUSE:0x08}}so I fig... you'd moved, {{PAUSE:0x08}}{{TAI... {{CONTINUE}}
1543	{{SET_DEMO_ORDER_NPC0:0x000... {{CLEAR}}{{SET_DEMO_ORDER_NP... to ol' {{PLAYER_NAME}}!{{PAUSE:0... Right, {{PAUSE:0x08}}{{TAIL}}?{{BU... {{CLEAR}}{{SET_DEMO_ORDER_NP... My instincts are the greatest in the ENTIRE UNIVERSE!{{PAUSE:... Yeah, {{PAUSE:0x08}}{{TAIL}}! {{CONTINUE}}
1544	Hey, {{PAUSE:0x08}}what's up with... this evening?{{BUTTON}} {{CLEAR}}{{SET_DEMO_ORDER_NP... I talked to you, huh,{{PAUSE:0x05}} {{TAIL}}?! {{CONTINUE}}
1545	How-diddly-do!{{PAUSE:0x14}} So you're sort of a loner,{{PAUSE:0... aren't you, {{PAUSE:0x08}}{{TAIL}}?... {{CLEAR}}You know, if you don't s... to me every now and again, {{SET_DEMO_ORDER_NPC0:0x000... {{TAIL}}!{{PAUSE:0x08}} No lie! {{CONTINUE}}
1546	{{SET_DEMO_ORDER_NPC0:0x000... If it isn't {{PLAYER_NAME}}!{{SET_D... {{CLEAR}}{{SET_DEMO_ORDER_NP... how I haven't seen you lately,{{PA... and here you are wandering

String table entry

ID 1546

Text

```
{{SET_DEMO_ORDER_NPC0:0x000003}}Hello, fellow night owl!{{PAUSE:0x0c}}
If it isn't {{PLAYER_NAME}}!{{SET_DEMO_ORDER_NPC0:0x000015}}{{BUTTON}}
{{CLEAR}}{{SET_DEMO_ORDER_NPC0:0x0000ff}}I was just thinking about
how I haven't seen you lately,{{PAUSE:0x08}}
and here you are wandering
in the dark, {{PAUSE:0x08}}{{TAIL}}!
{{CONTINUE}}
```



ID	Content
	{{CONTINUE}}
1542	Hey, {{PAUSE:0x08}}{{PLAYER_NA... {{CLEAR}}I haven't seen you around these days, {{PAUSE:0x08}}so I fig... you'd moved, {{PAUSE:0x08}}{{TAI... {{CONTINUE}}
1543	{{SET_DEMO_ORDER_NPC0:0x000... {{CLEAR}}{{SET_DEMO_ORDER_NP... to ol' {{PLAYER_NAME}}!{{PAUSE:0... Right, {{PAUSE:0x08}}{{TAIL}}?{{BU... {{CLEAR}}{{SET_DEMO_ORDER_NP... My instincts are the greatest in the ENTIRE UNIVERSE!{{PAUSE:... Yeah, {{PAUSE:0x08}}{{TAIL}}! {{CONTINUE}}
1544	Hey, {{PAUSE:0x08}}what's up with... this evening?{{BUTTON}} {{CLEAR}}{{SET_DEMO_ORDER_NP... I talked to you, huh,{{PAUSE:0x05}} {{TAIL}}?! {{CONTINUE}}
1545	How-diddly-do!{{PAUSE:0x14}} So you're sort of a loner,{{PAUSE:0... aren't you, {{PAUSE:0x08}}{{TAIL}}?... {{CLEAR}}You know, if you don't s... to me every now and again, {{SET_DEMO_ORDER_NPC0:0x000... {{TAIL}}!{{PAUSE:0x08}} No lie! {{CONTINUE}}
1546	{{SET_DEMO_ORDER_NPC0:0x000... If it isn't {{PLAYER_NAME}}!{{SET_D... {{CLEAR}}{{SET_DEMO_ORDER_NP... how I haven't seen you lately,{{PA... and here you are wandering in the dark, {{PAUSE:0x08}}{{TAIL}}!

Msg #1543 | Msg #1545 | Msg #1546 ✕

String table entry

ID: 1546

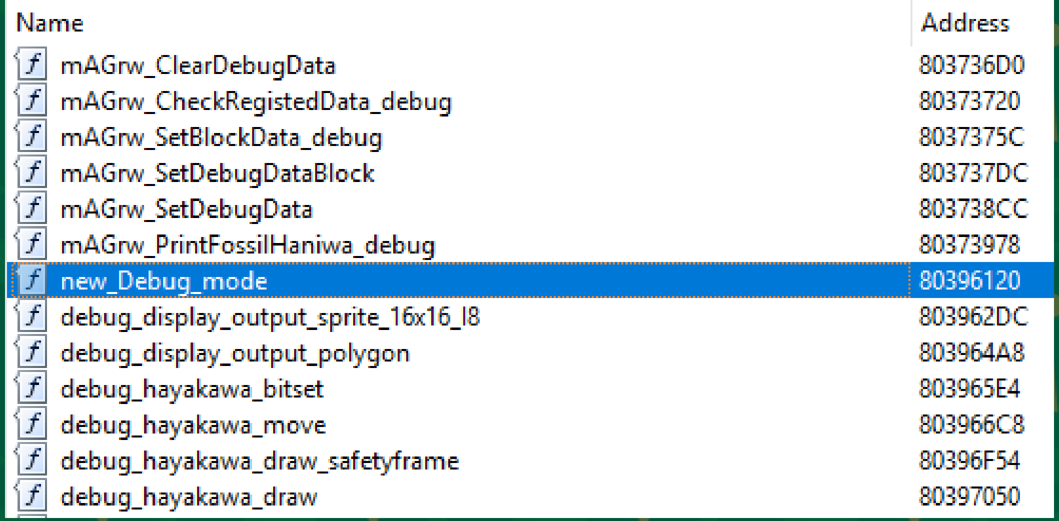
Text:

```
{{SET_DEMO_ORDER_NPC0:0x000003}}Hello, fellow night owl!{{PAUSE:0x0c}}
If it isn't {{PLAYER_NAME}}!{{SET_DEMO_ORDER_NPC0:0x000015}}{{BUTTON}}
{{CLEAR}}{{SET_DEMO_ORDER_NPC0:0x0000ff}}I was just thinking about
how I haven't seen you lately,{{PAUSE:0x08}}
and here you are wandering
in the dark, {{PAUSE:0x08}}{{TAIL}}!
{{CONTINUE}}
```

Unlocking Developer Features

Finding debug features

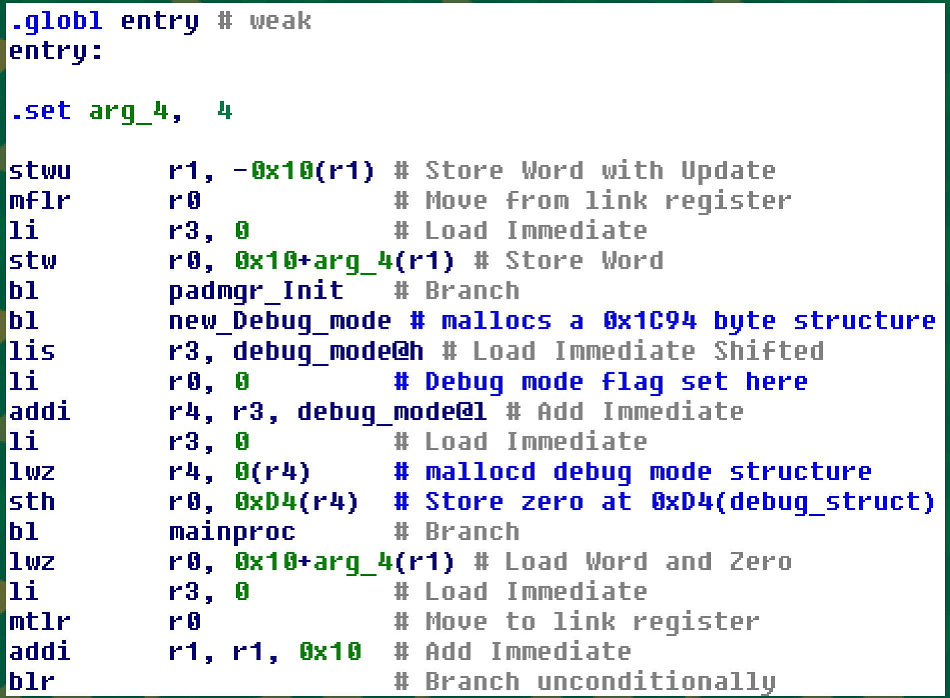
- Noticed a bunch of functions and variables with “debug” in the name
- Debug features would be useful for testing out mods
- What does `new_Debug_mode` do?



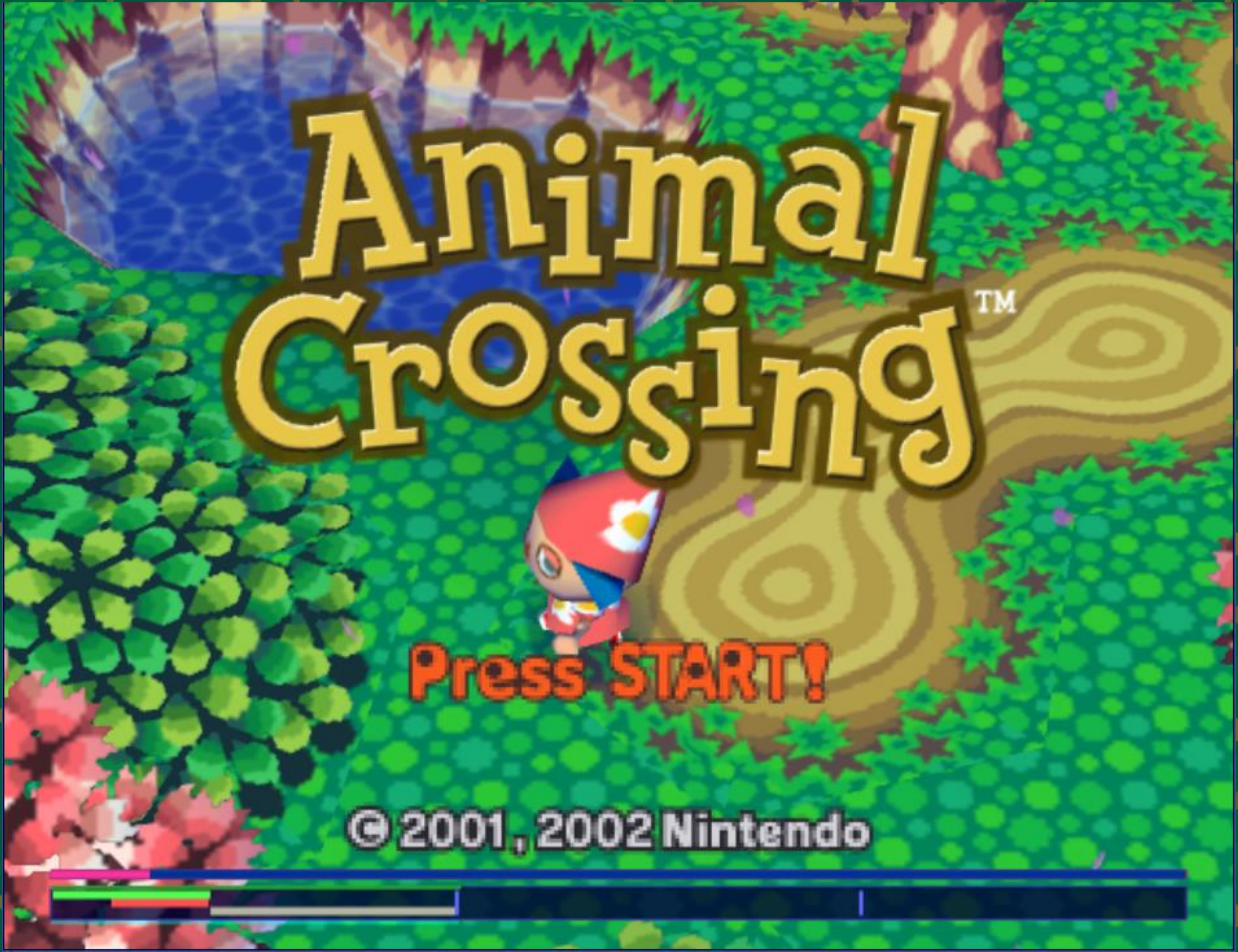
Name	Address
mAGrw_ClearDebugData	803736D0
mAGrw_CheckRegistedData_debug	80373720
mAGrw_SetBlockData_debug	8037375C
mAGrw_SetDebugDataBlock	803737DC
mAGrw_SetDebugData	803738CC
mAGrw_PrintFossilHaniwa_debug	80373978
new_Debug_mode	80396120
debug_display_output_sprite_16x16_I8	803962DC
debug_display_output_polygon	803964A8
debug_hayakawa_bitset	803965E4
debug_hayakawa_move	803966C8
debug_hayakawa_draw_safetyframe	80396F54
debug_hayakawa_draw	80397050

Finding debug features

- Called by `entry` (right after the Nintendo logo splashscreen)
- Allocates a `0x1C94` byte structure and saves the pointer to it
- Value at offset `0xD4` is set to zero right away
- What happens if it's set to 1?



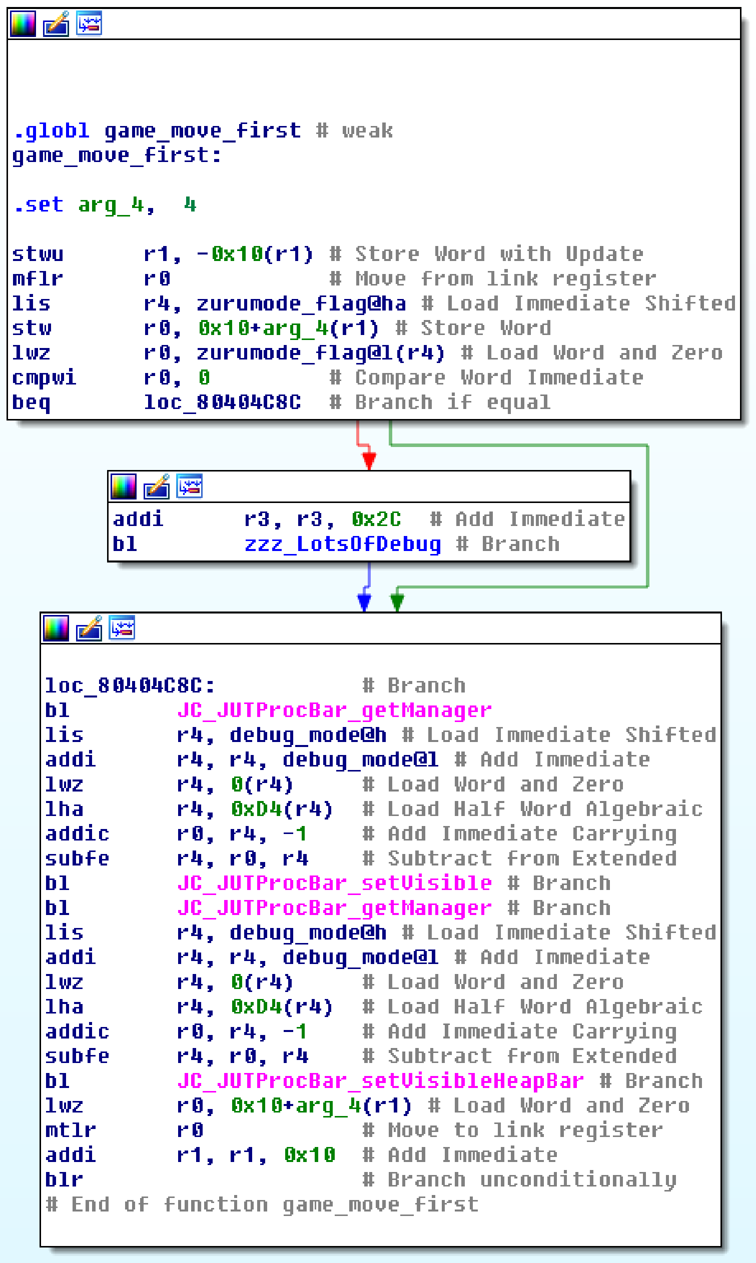
```
.globl entry # weak
entry:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mflr      r0            # Move from link register
li        r3, 0         # Load Immediate
stw       r0, 0x10+arg_4(r1) # Store Word
bl        padmgr_Init   # Branch
bl        new_Debug_mode # mallocs a 0x1C94 byte structure
lis       r3, debug_mode@h # Load Immediate Shifted
li        r0, 0         # Debug mode flag set here
addi      r4, r3, debug_mode@l # Add Immediate
li        r3, 0         # Load Immediate
lwz       r4, 0(r4)     # mallocd debug mode structure
sth       r0, 0xD4(r4)  # Store zero at 0xD4(debug_struct)
bl        mainproc      # Branch
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
li        r3, 0         # Load Immediate
mtlr      r0            # Move to link register
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
```



Animal Crossing™
Press START!
© 2001, 2002 Nintendo

Finding debug features

- Looking for more code that references the debug mode structure…
- There are a bunch of references to “zuru mode” in the context of debug display behavior
  - No idea what it is or what “zuru” means (zulu?)
- `zurumode_flag` looks important



```
.globl game_move_first # weak
game_move_first:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mflr      r0            # Move from link register
lis       r4, zurumode_flag@ha # Load Immediate Shifted
stw       r0, 0x10+arg_4(r1) # Store Word
lwz       r0, zurumode_flag@l(r4) # Load Word and Zero
cmpwi     r0, 0         # Compare Word Immediate
beq       loc_80404C8C  # Branch if equal
```

```
addi      r3, r3, 0x2C  # Add Immediate
bl        zzz_LotsOfDebug # Branch
```

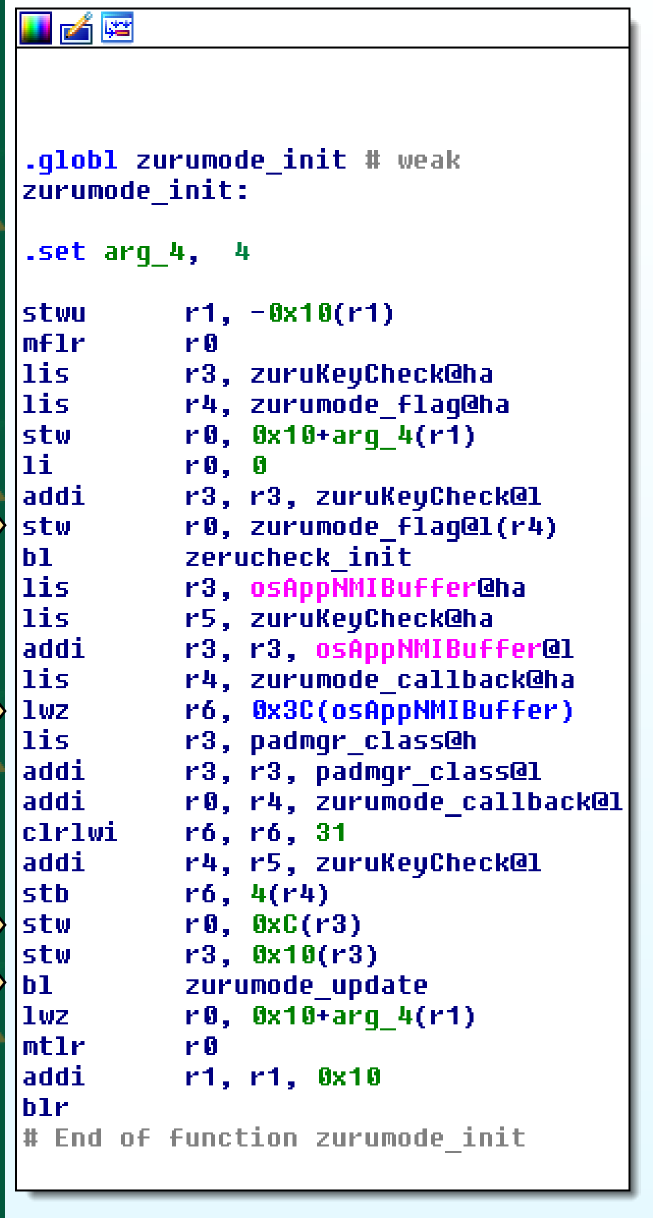
```
loc_80404C8C:           # Branch
bl        JC_JUTProcBar_getManager
lis       r4, debug_mode@h # Load Immediate Shifted
addi      r4, r4, debug_mode@l # Add Immediate
lwz       r4, 0(r4)     # Load Word and Zero
lha       r4, 0xD4(r4)  # Load Half Word Algebraic
addic     r0, r4, -1    # Add Immediate Carrying
subfe     r4, r0, r4    # Subtract from Extended
bl        JC_JUTProcBar_setVisible # Branch
bl        JC_JUTProcBar_getManager # Branch
lis       r4, debug_mode@h # Load Immediate Shifted
addi      r4, r4, debug_mode@l # Add Immediate
lwz       r4, 0(r4)     # Load Word and Zero
lha       r4, 0xD4(r4)  # Load Half Word Algebraic
addic     r0, r4, -1    # Add Immediate Carrying
subfe     r4, r0, r4    # Subtract from Extended
bl        JC_JUTProcBar_setVisibleHeapBar # Branch
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
mtlr      r0            # Move to link register
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
# End of function game_move_first
```

Finding debug features

- Looked up functions with zurumode in the name:
  - zurumode_init
  - zurumode_callback
  - zurumode_update
  - zurumode_cleanup

zurumode_init

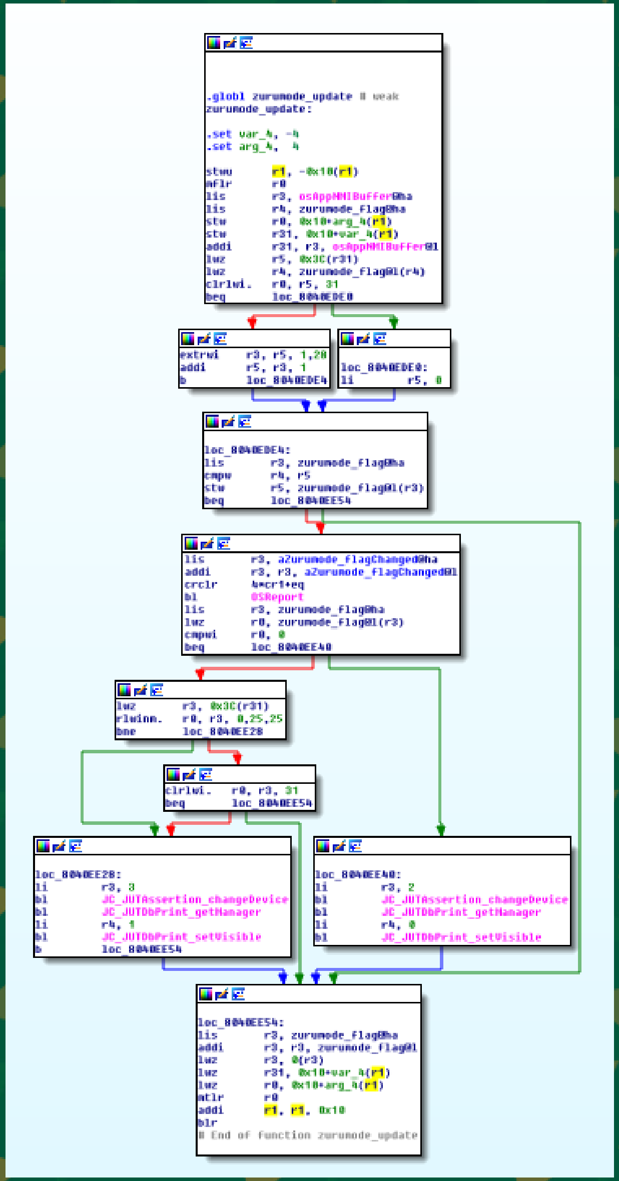
- Sets zurumode_flag to 0
- Checks some bits in a thing called osAppNMIBuffer
- Stores pointer to zurumode_callback in padmgr structure
- Calls zurumode_update



```
.globl zurumode_init # weak
zurumode_init:

.set arg_4,  4

stwu      r1, -0x10(r1)
mflr      r0
lis       r3, zuruKeyCheck@ha
lis       r4, zurumode_flag@ha
stw       r0, 0x10+arg_4(r1)
li        r0, 0
addi      r3, r3, zuruKeyCheck@l
stw       r0, zurumode_flag@l(r4)
bl        zerucheck_init
lis       r3, osAppNMIBuffer@ha
lis       r5, zuruKeyCheck@ha
addi      r3, r3, osAppNMIBuffer@l
lis       r4, zurumode_callback@ha
lwz       r6, 0x3C(osAppNMIBuffer)
lis       r3, padmgr_class@h
addi      r3, r3, padmgr_class@l
addi      r0, r4, zurumode_callback@l
clrlwi    r6, r6, 31
addi      r4, r5, zuruKeyCheck@l
stb       r6, 4(r4)
stw       r0, 0xC(r3)
stw       r3, 0x10(r3)
bl        zurumode_update
lwz       r0, 0x10+arg_4(r1)
mtlr      r0
addi      r1, r1, 0x10
blr
# End of function zurumode_init
```

zurumode_update

- Checks some bits in `osAppNMIBuffer`.
- Conditionally update the value of `zurumode_flag` based on the bits
- Prints a format string to OS debug console
- Characters are not ASCII, so I tried Japanese encodings. It’s Shift-JIS:
  - “zurumode_flag が %d から %d に変更されました”
  - “zurumode_flag has been changed from %d to %d”
  - Doesn’t mean much yet, but knowing the encoding helped with other debug strings and untranslated game text



```
.globl zurumode_update # weak
zurumode_update:

.set var_4, -4
.set arg_4,  4

stwu      r1, -0x10(r1)
mflr      r0
lis       r3, osAppNMIBuffer@ha
lis       r4, zurumode_flag@ha
stw       r0, 0x10+arg_4(r1)
stw       r31, 0x10+var_4(r1)
addi      r31, r3, osAppNMIBuffer@l
lwz       r5, 0x3C(r31)
lwz       r4, zurumode_flag@l(r4)
clrlwi.   r0, r5, 31
beq       loc_8040EDE0
```

```
extrwi    r3, r5, 1,28
addi      r5, r3, 1
b         loc_8040EDE4
```

```
loc_8040EDE0:
li            r5, 0
```

```
loc_8040EDE4:
lis       r3, zurumode_flag@ha
cmpw      r4, r5
stw       r5, zurumode_flag@l(r3)
beq       loc_8040EE54
```

```
lis       r3, aZurumode_flagChanged@ha
addi      r3, r3, aZurumode_flagChanged@l
crclr     4*cr1+eq
bl        OSReport
lis       r3, zurumode_flag@ha
lwz       r0, zurumode_flag@l(r3)
cmpwi     r0, 0
beq       loc_8040EE40
```

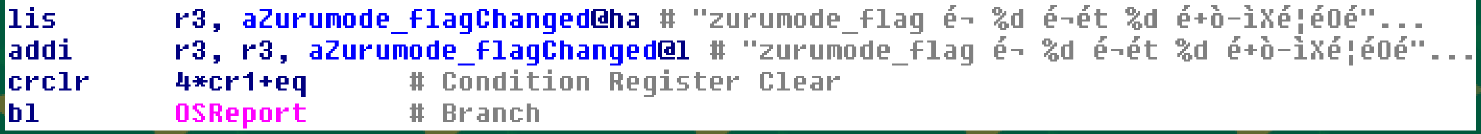
```
lwz       r3, 0x3C(r31)
rlwinm.   r0, r3, 0,25,25
bne       loc_8040EE28
```

```
clrlwi.   r0, r3, 31
beq       loc_8040EE54
```

```
loc_8040EE28:
li        r3, 3
bl        JC_JUTAssertion_changeDevice
bl        JC_JUTDbPrint_getManager
li        r4, 1
bl        JC_JUTDbPrint_setVisible
b         loc_8040EE54
```

```
loc_8040EE40:
li        r3, 2
bl        JC_JUTAssertion_changeDevice
bl        JC_JUTDbPrint_getManager
li        r4, 0
bl        JC_JUTDbPrint_setVisible
```

```
loc_8040EE54:
lis       r3, zurumode_flag@ha
addi      r3, r3, zurumode_flag@l
lwz       r3, 0(r3)
lwz       r31, 0x10+var_4(r1)
lwz       r0, 0x10+arg_4(r1)
mtlr      r0
addi      r1, r1, 0x10
blr
# End of function zurumode_update
```

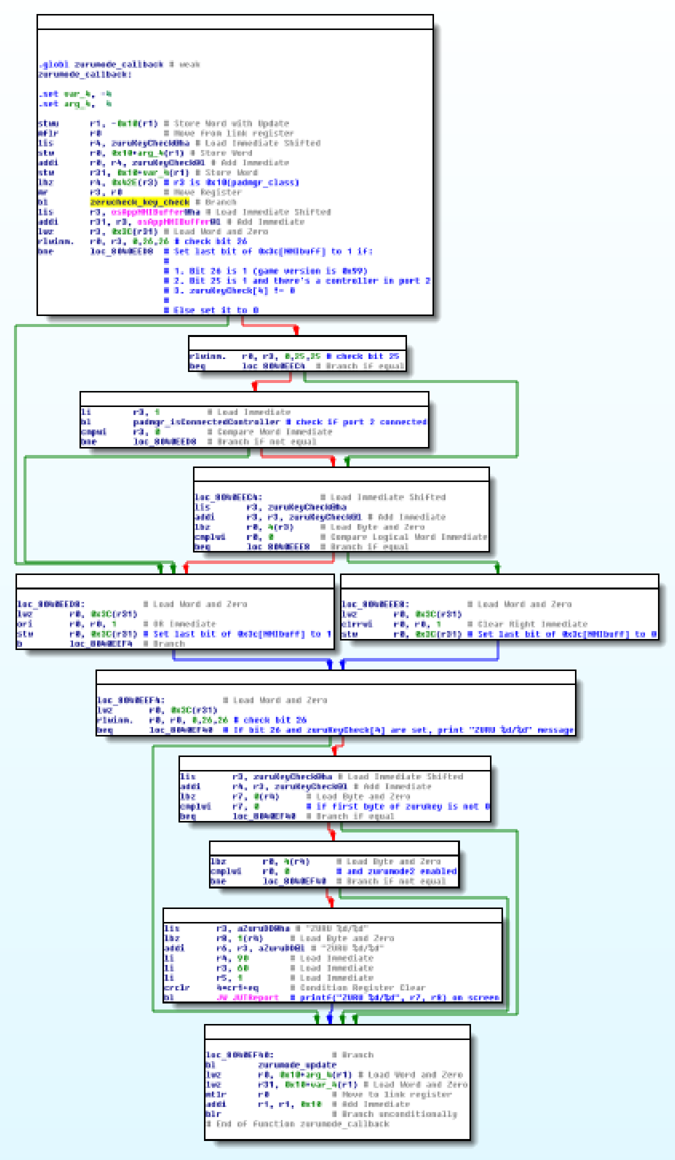


```
lis         r3, aZurumode_flagChanged@ha # "zurumode_flag é¬ %d é¬ét %d é+ò-ìXé¦éOé"...
addi        r3, r3, aZurumode_flagChanged@l # "zurumode_flag é¬ %d é¬ét %d é+ò-ìXé¦éOé"...
crclr       4*cr1+eq        # Condition Register Clear
bl          OSReport        # Branch
```

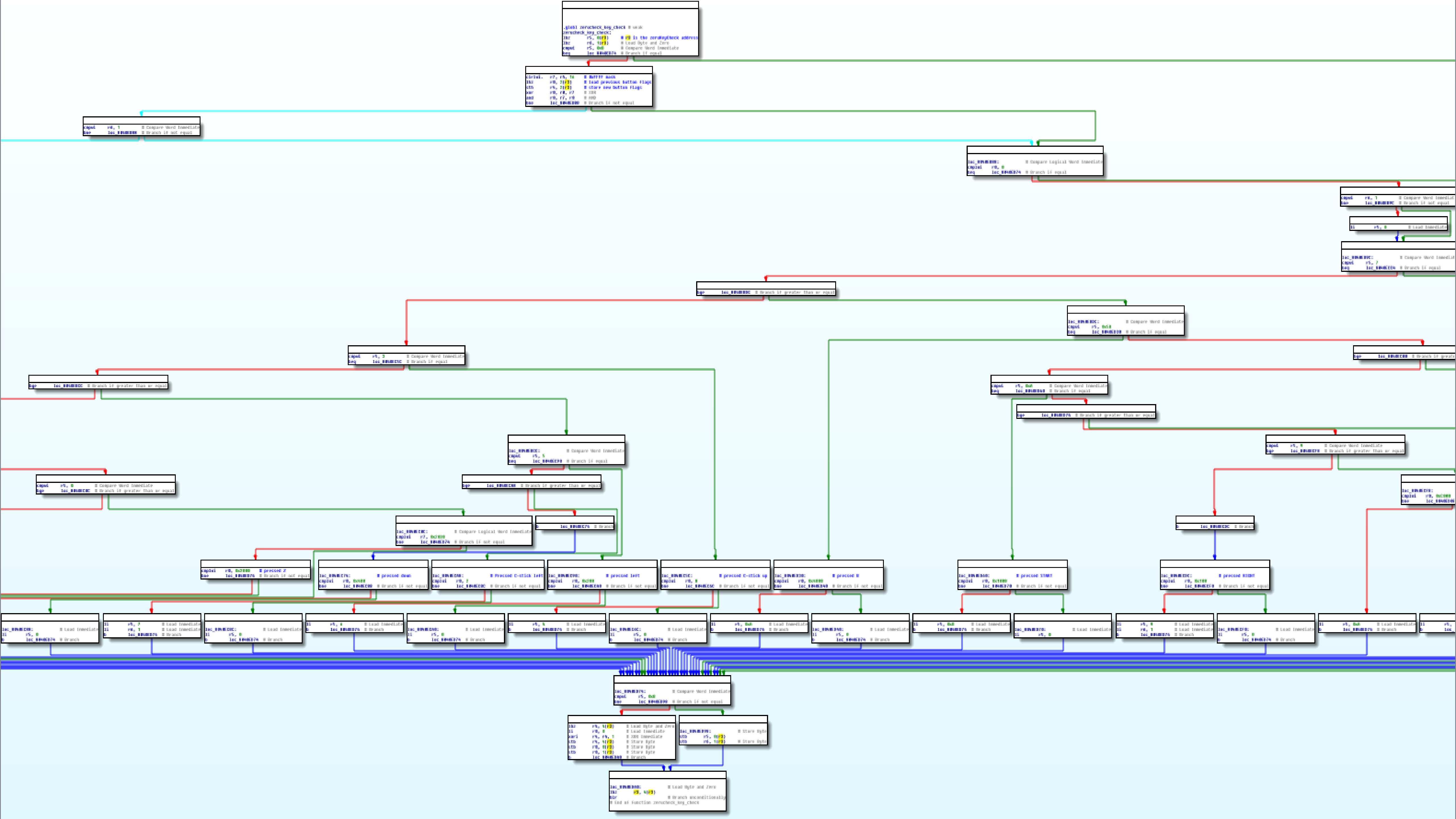
zurumode_callback

- Calls `zerumode_check_keycheck` first
- Checks a bunch of bits in `osAppNMIBuffer`
- Prints value of zuru mode flag
- Calls `zurumode_update`

…what's `zerumode_check_keycheck`?



```
.globl zurumode_callback # weak
zurumode_callback:

.set var_4, -4
.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mflr      r0            # Move from link register
lis       r4, zuruKeyCheck@ha # Load Immediate Shifted
stw       r0, 0x10+arg_4(r1) # Store Word
addi      r0, r4, zuruKeyCheck@l # Add Immediate
stw       r31, 0x10+var_4(r1) # Store Word
lbz       r4, 0x426(r3) # r3 is 0x10(padmgr_class)
mr        r3, r0        # Move Register
bl        zerucheck_key_check # Branch
lis       r3, osAppNMIBuffer@ha # Load Immediate Shifted
addi      r31, r3, osAppNMIBuffer@l # Add Immediate
lwz       r3, 0x3C(r31) # Load Word and Zero
rlwinm.   r0, r3, 0,26,26 # check bit 26
bne       loc_8040EED8  # Set last bit of 0x3c[NMIbuff] to 1 if:
                        #
                        # 1. Bit 26 is 1 (game version is 0x99)
                        # 2. Bit 25 is 1 and there's a controller in port 2
                        # 3. zuruKeyCheck[4] != 0
                        #
                        # Else set it to 0

rlwinm.   r0, r3, 0,25,25 # check bit 25
beq       loc_8040EEC4  # Branch if equal

li        r3, 1         # Load Immediate
bl        padmgr_isConnectedController # check if port 2 connected
cmpwi     r3, 0         # Compare Word Immediate
bne       loc_8040EED8  # Branch if not equal

loc_8040EEC4:           # Load Immediate Shifted
lis       r3, zuruKeyCheck@ha
addi      r3, r3, zuruKeyCheck@l # Add Immediate
lbz       r0, 4(r3)     # Load Byte and Zero
cmplwi    r0, 0         # Compare Logical Word Immediate
beq       loc_8040EEE8  # Branch if equal

loc_8040EED8:           # Load Word and Zero
lwz       r0, 0x3C(r31)
ori       r0, r0, 1     # OR Immediate
stw       r0, 0x3C(r31) # Set last bit of 0x3c[NMIbuff] to 1
b         loc_8040EEF4  # Branch

loc_8040EEE8:           # Load Word and Zero
lwz       r0, 0x3C(r31)
clrrwi    r0, r0, 1     # Clear Right Immediate
stw       r0, 0x3C(r31) # Set last bit of 0x3c[NMIbuff] to 0

loc_8040EEF4:           # Load Word and Zero
lwz       r0, 0x3C(r31)
rlwinm.   r0, r0, 0,26,26 # check bit 26
beq       loc_8040EF40  # If bit 26 and zuruKeyCheck[4] are set, print "ZURU %d/%d" message

lis       r3, zuruKeyCheck@ha # Load Immediate Shifted
addi      r4, r3, zuruKeyCheck@l # Add Immediate
lbz       r7, 0(r4)     # Load Byte and Zero
cmplwi    r7, 0         # if first byte of zurukey is not 0
beq       loc_8040EF40  # Branch if equal

lbz       r0, 4(r4)     # Load Byte and Zero
cmplwi    r0, 0         # and zurumode2 enabled
bne       loc_8040EF40  # Branch if not equal

lis       r3, aZuruDD@ha # "ZURU %d/%d"
lbz       r8, 1(r4)     # Load Byte and Zero
addi      r6, r3, aZuruDD@l # "ZURU %d/%d"
li        r4, 90        # Load Immediate
li        r3, 60        # Load Immediate
li        r5, 1         # Load Immediate
crclr     4*cr1+eq      # Condition Register Clear
bl        JW_JUTReport  # printf("ZURU %d/%d", r7, r8) on screen

loc_8040EF40:           # Branch
bl        zurumode_update
lwz       r0, 0x10+arg_4(r1) # Load Word and Zero
lwz       r31, 0x10+var_4(r1) # Load Word and Zero
mtlr      r0            # Move to link register
addi      r1, r1, 0x10  # Add Immediate
blr                     # Branch unconditionally
# End of function zurumode_callback
```



```
.globl zerucheck_key_check # weak
zerucheck_key_check:
lbz       r5, 0(r3)       # r3 is the zeruKeyCheck address
lbz       r6, 1(r3)       # Load Byte and Zero
cmpwi     r5, 0x0         # Compare Word Immediate
beq       loc_[illegible] # Branch if equal
```

```
clrlwi.   r7, r4, 16      # 0xFFFF mask
lhz       r0, 2(r3)       # load previous button flags
sth       r4, 2(r3)       # store new button flags
xor       r0, r0, r7      # XOR
and       r0, r7, r0      # AND
bne       loc_[illegible] # Branch if not equal
```

```
cmpwi     r6, 1           # Compare Word Immediate
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # Compare Logical Word Immediate
cmplwi    r0, 0
beq       loc_[illegible] # Branch if equal
```

```
loc_[illegible]:          # Compare Logical Word Immediate
cmplwi    r7, 0x2020
bne       loc_[illegible] # Branch if not equal
```

```
cmplwi    r0, 0x2000      # precoded Z
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded down
cmplwi    r0, 0x400
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # Precoded C-stick left
cmplwi    r0, 2
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded left
cmplwi    r0, 0x200
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded C-stick up
cmplwi    r0, 8
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded B
cmplwi    r0, 0x800
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded START
cmplwi    r0, 0x1000
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # precoded RIGHT
cmplwi    r0, 0x100
bne       loc_[illegible] # Branch if not equal
```

```
loc_[illegible]:          # Compare Word Immediate
cmpwi     r5, 0x0
bne       loc_[illegible] # Branch if not equal
```

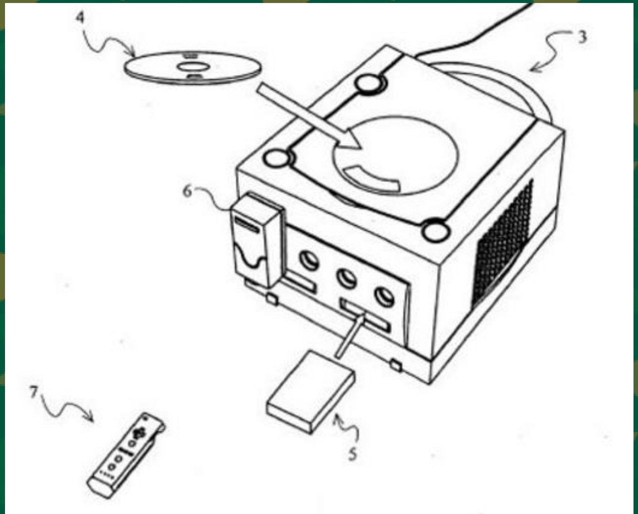
```
lbz       r4, 4(r3)       # Load Byte and Zero
li        r0, 0           # Load Immediate
xori      r4, r4, 1       # XOR Immediate
stb       r4, 4(r3)       # Store Byte
stb       r0, 0(r3)       # Store Byte
stb       r0, 1(r3)       # Store Byte
b         loc_[illegible] # Branch
```

```
loc_[illegible]:          # Store Byte
stb       r5, 0(r3)
stb       r6, 1(r3)       # Store Byte
```

```
loc_[illegible]:          # Load Byte and Zero
lbz       r3, 4(r3)
blr                       # Branch unconditionally
# End of function zerucheck_key_check
```

zerumode_check_keycheck

- Didn’t know what zuru mode was or how crucial it was to debugging
- Tried getting translations of “zuru” or slight changes in spelling through Google Translate, got “shake”
  - Thought this might refer to original Wii remote, which was actually made for Gamecube, or some other special kind of developer input device
- Didn’t know if “key check” referred to cryptographic key, controller buttons, or keyboard keys
- Noticed missing/inconsistent symbols
- Held off and looked for path of least resistance

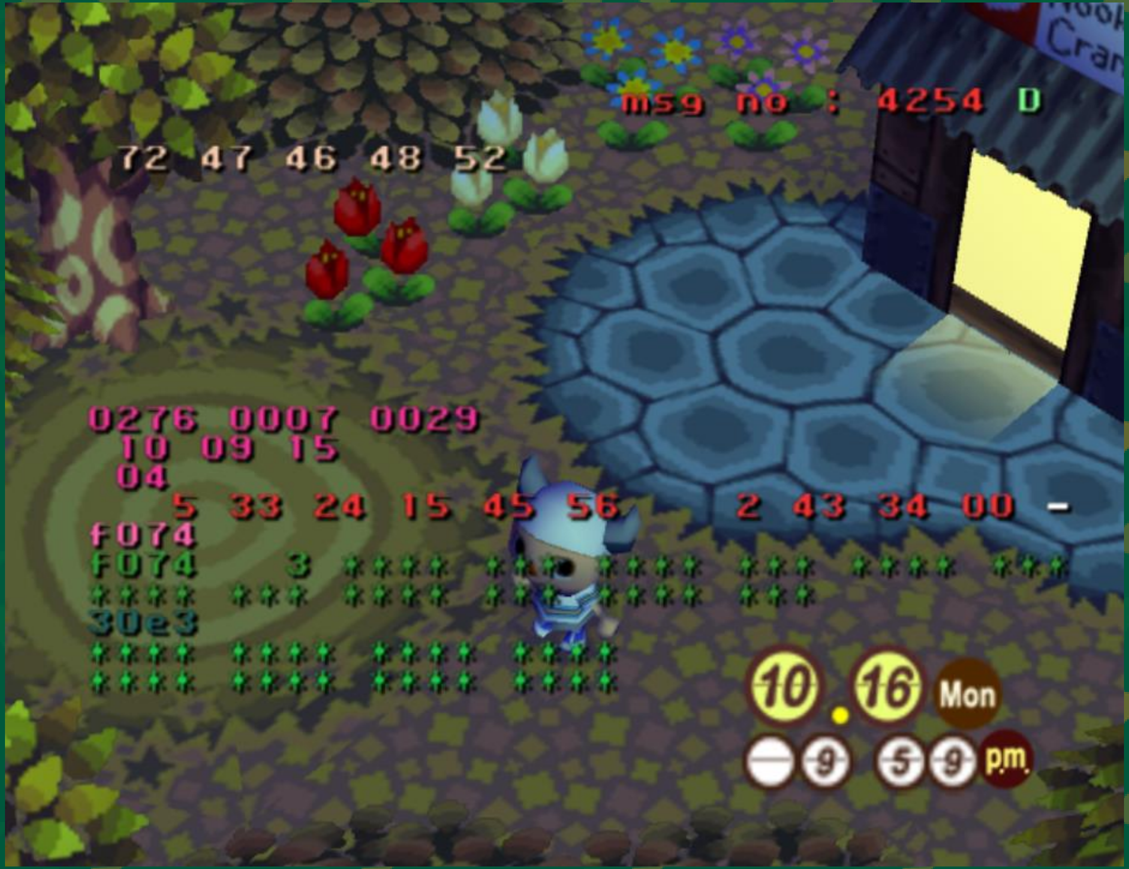


Finding debug features

- Problem with the symbol loader
  - First script parsed out address/name and added it
  - Section addresses in `foresta` map all start from 0
    - Resulted in symbols clobbering each other
- New scripts set up values for each section with:
  - Name at correct address
  - Function or data definition
  - Segment named after the source object (e.g. `m_player_lib.o`)
- The new `bss` segment for `m_debug_mode.o` had some variables like `quest_draw_status` and `event_status`.
  - Cross-references from these data entries to a huge piece of code that checks `debug_print_flg` (located in the same `bss` segment)

Finding debug features

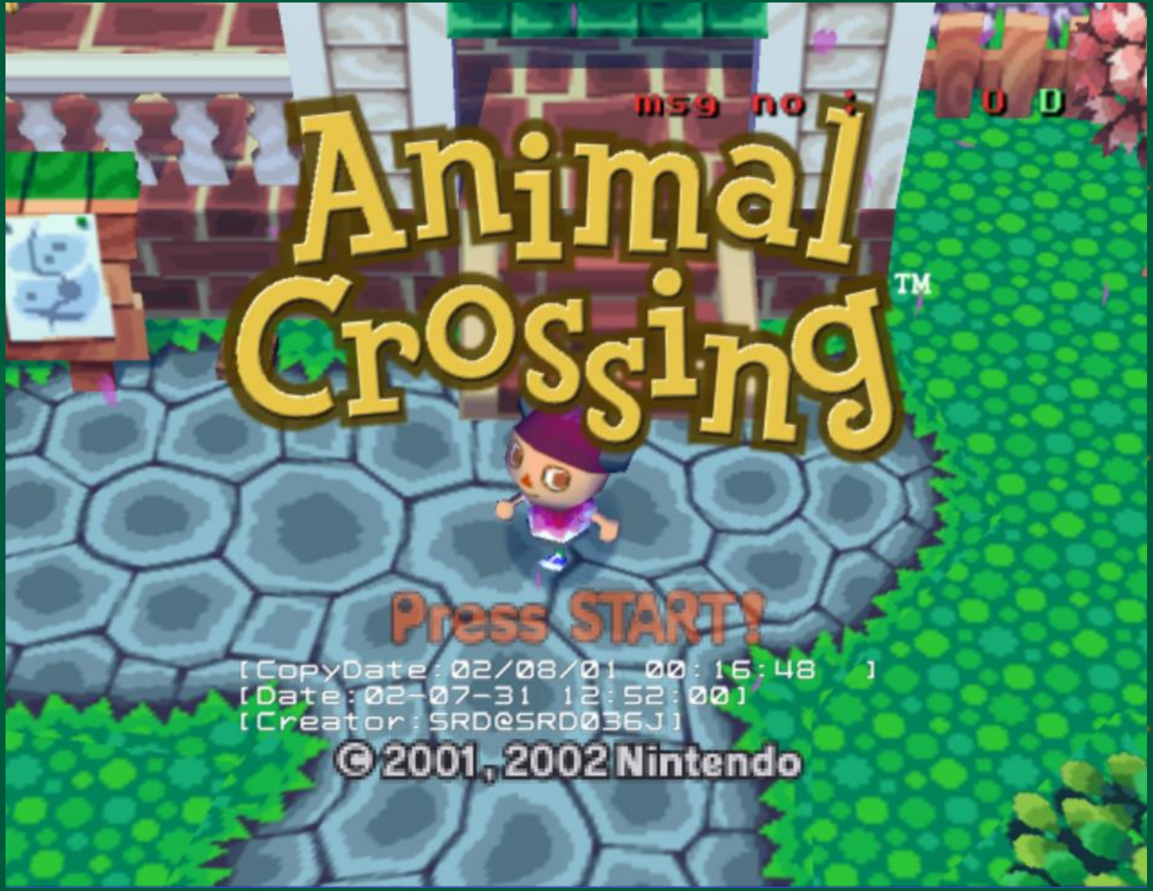
- Approach in reverse
  - Go from debug display behaviors back up to debug mode activation
- Did some simple NOPping to bypass checks and get displays to activate
- Found `debug_print_flg` and some status variables related to it
  - Set breakpoint where `debug_print_flg` is checked. Never hits.
  - Why? `zurumode_flag` gets checked first.
- `zurumode_flag` keeps showing up throughout debug code and simple patches get the various displays to activate
  - No avoiding it any longer, I have to figure out what zuru mode is



```
                                         msg no : 4254 D
  72 47 46 48 52

0276 0007 0029
 10 09 15
 04
   5 33 24 15 45 56      2 43 34 00 -
f074
F074     3 **** *** **** *** **** ***
**** *** **** *** **** ***
30e3
**** **** **** ****
**** **** **** ****
```

`zurumode_init`

- Returning to `zurumode_init`, it initializes a few things:
  - `0xC(padmgr_class)` is set to the address of `zurumode_callback`
  - `0x4(zuruKeyCheck)` is set to the last bit of the 32-bit value at `0x3C(osAppNMIBuffer)`
- Only runs once on game start
- Patching it to set `0x4(zuruKeyCheck)` to 1 causes this text to appear on the title screen
  - But none of the other displays show up during play



```
msg no :
0 D
[CopyDate:02/08/01 00:16:48   ]
[Date:02-07-31 12:52:00]
[Creator:SRD@SRD036J]
```

zurumode_update

- Checks the last bit of 0x3C (osAppNMIBuffer) and updates zurumode_flag based on its value
  - If it's zero, the flag is set to zero.
  - If not, it extracts bit 28 from the NMI buffer value and adds 1 to it
    - The result will always be 1 or 2. The flag is set to this value.
    - When the result is 2 a bunch of interesting stuff shows up.
- Checks whether the flag has changed
  - If so, it calls some functions from boot.dol:

```
manager = JC_JUTDbPrint_getManager()
if (flag == 0) {
    JC_JUTAssertion_changeDevice(2)
    JC_JUTDbPrint_setVisible(manager, 0)
} else if (BIT(nmiBuf+0x3c, 25) || BIT(nmiBuf+0x3c, 31)) {
    JC_JUTAssertion_changeDevice(3)
    JC_JUTDbPrint_setVisible(manager, 1)
}
```

`zurumode_callback`

- Runs each time the gamepad state updates
- Calls the crazy `zerumode_check_keycheck` function
- Checks and sets some bits in `0x3c` (`osAppNMIBuffer`)
- Calls `zurumode_update`
- *The last bit of the NMI buffer value is set if:*
  - bit 26 is set, or…
  - bit 25 is set and controller 2 is plugged in, or…
  - `0x4` (`zuruKeyCheck`) is non-zero
- Otherwise, the bit is set to zero (disabling zuru mode)

Enables zuru mode

Zuru Mode Activation

`0x3C(osAppNMIBuffer)`
Bit 26

?

`0x3C(osAppNMIBuffer)`
Bit 25

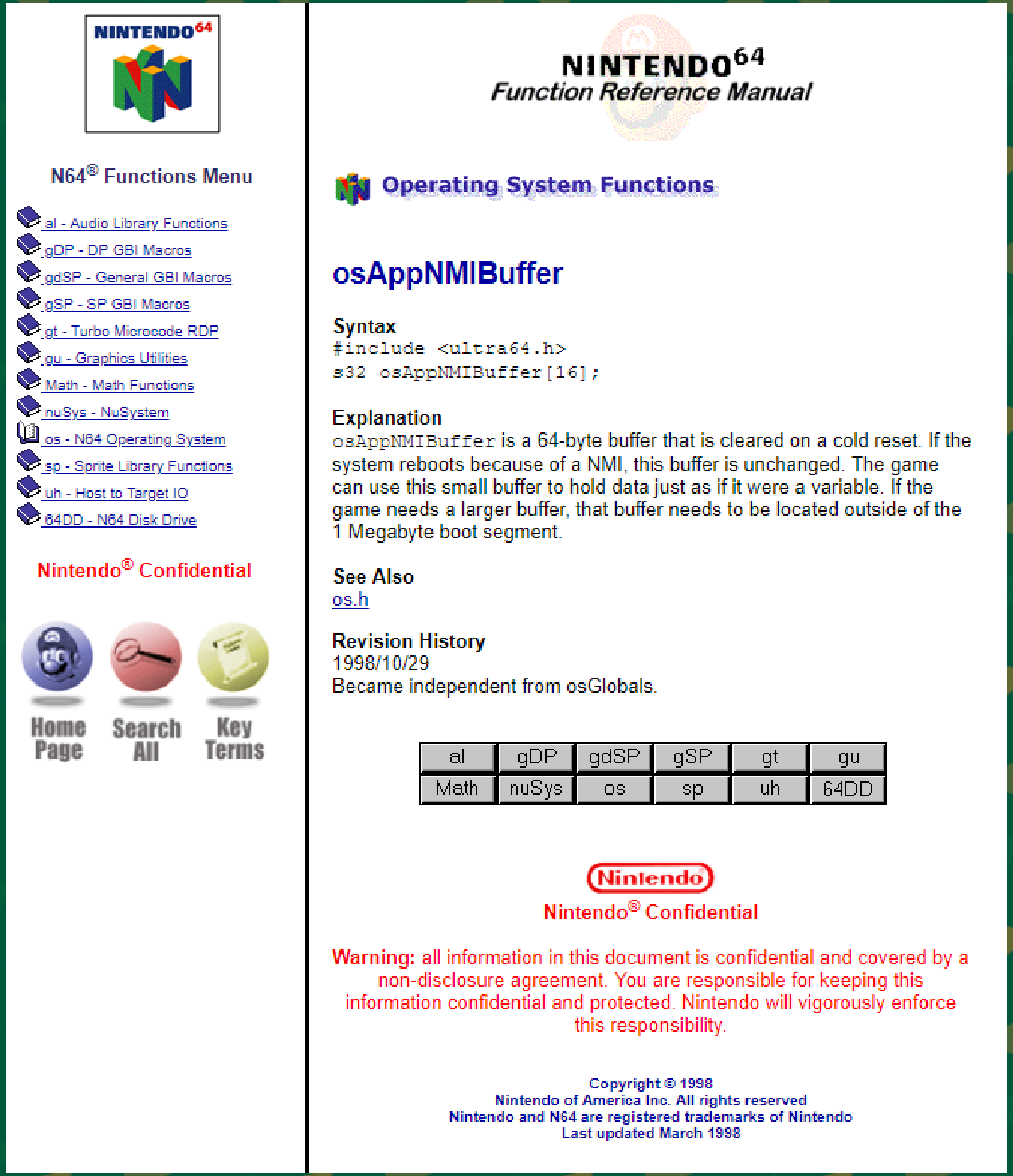
?

`0x4(zuruKeyCheck)`

?

osAppNMIBuffer

- What is `osAppNMIBuffer`?
- Found it in N64 SDK docs
  - "osAppNMIBuffer is a 64-byte buffer that is cleared on a cold reset. If the system reboots because of a NMI, this buffer is unchanged."
- NMI refers to soft reset (via non-maskable interrupt).
- Where do the bits get set?



NINTENDO64
Function Reference Manual

Operating System Functions

osAppNMIBuffer

Syntax

```
#include <ultra64.h>
s32 osAppNMIBuffer[16];
```

Explanation

`osAppNMIBuffer` is a 64-byte buffer that is cleared on a cold reset. If the system reboots because of a NMI, this buffer is unchanged. The game can use this small buffer to hold data just as if it were a variable. If the game needs a larger buffer, that buffer needs to be located outside of the 1 Megabyte boot segment.

See Also

os.h

Revision History

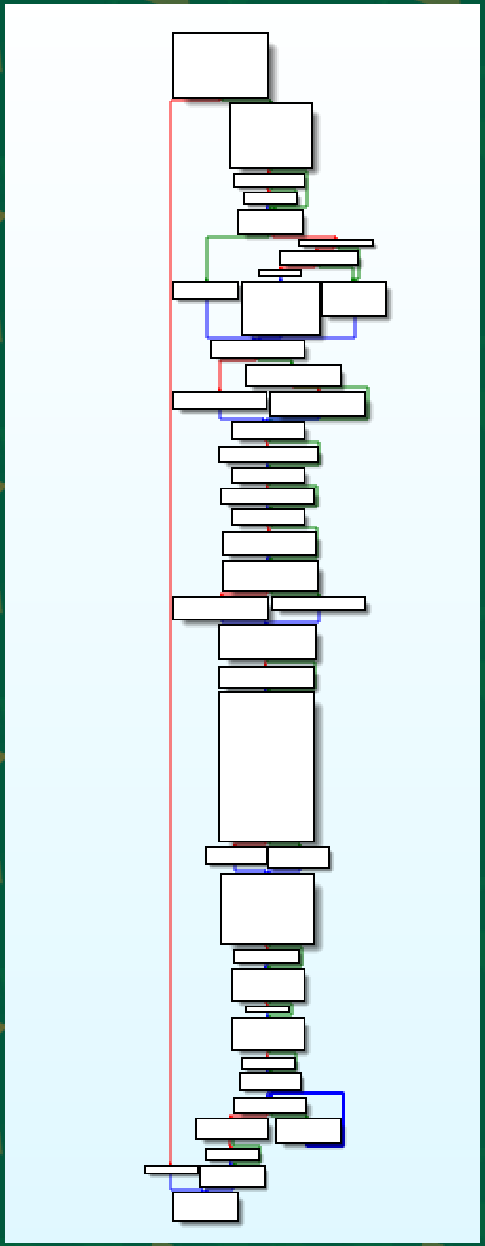
1998/10/29
Became independent from osGlobals.

Nintendo® Confidential

Warning: all information in this document is confidential and covered by a non-disclosure agreement. You are responsible for keeping this information confidential and protected. Nintendo will vigorously enforce this responsibility.

osAppNMIBuffer

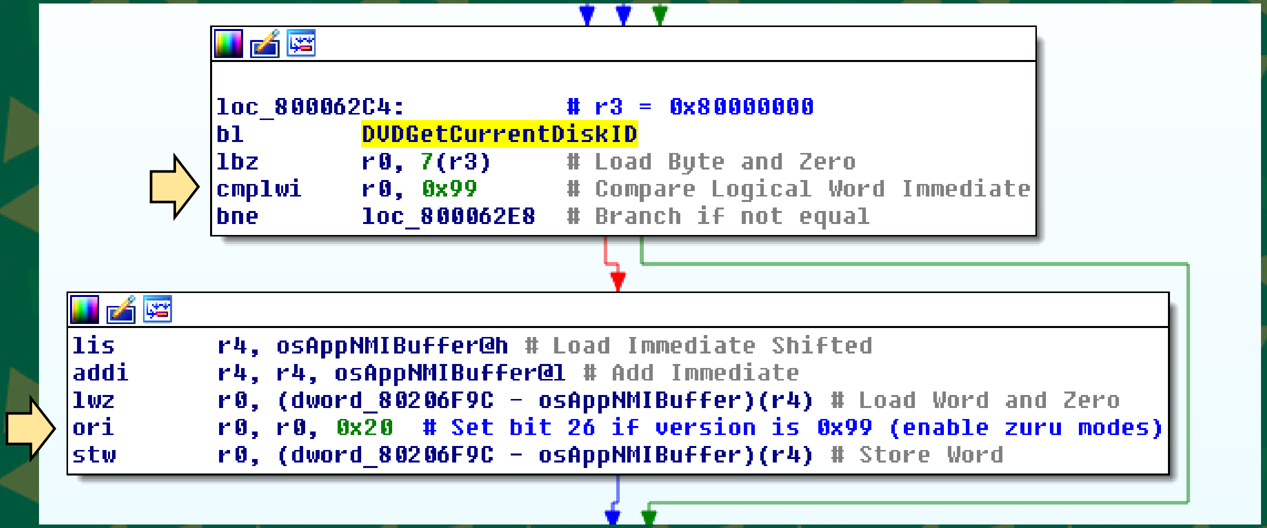
- Bits 25, 26, 28, and 31 of `0x3c(osAppNMIBuffer)` control zuru mode
  - 25 and 26 control whether it's enabled
  - 28 controls the flag level (1 or 2)
- A series of checks in the `main` function of `boot.dol` set bits in `osAppNMIBuffer`
  - Large, somewhat complex function
  - Look for OR instructions with 0x1, 0x8, 0x20, 0x40



boot.dol main function

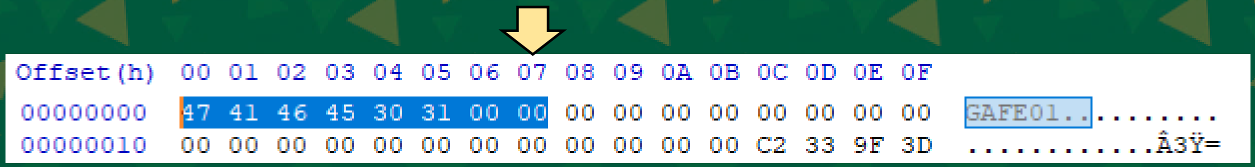
Bit 26

- First up: there's an `ori r0, r0, 0x20` instruction
  - Applied to the buffer value at 0x3c
  - Sets bit 26, which always results in zuru mode being enabled.
- To reach this block, the eighth byte of the disk ID must be 0x99
  - Try a simple patch for it in emulator...

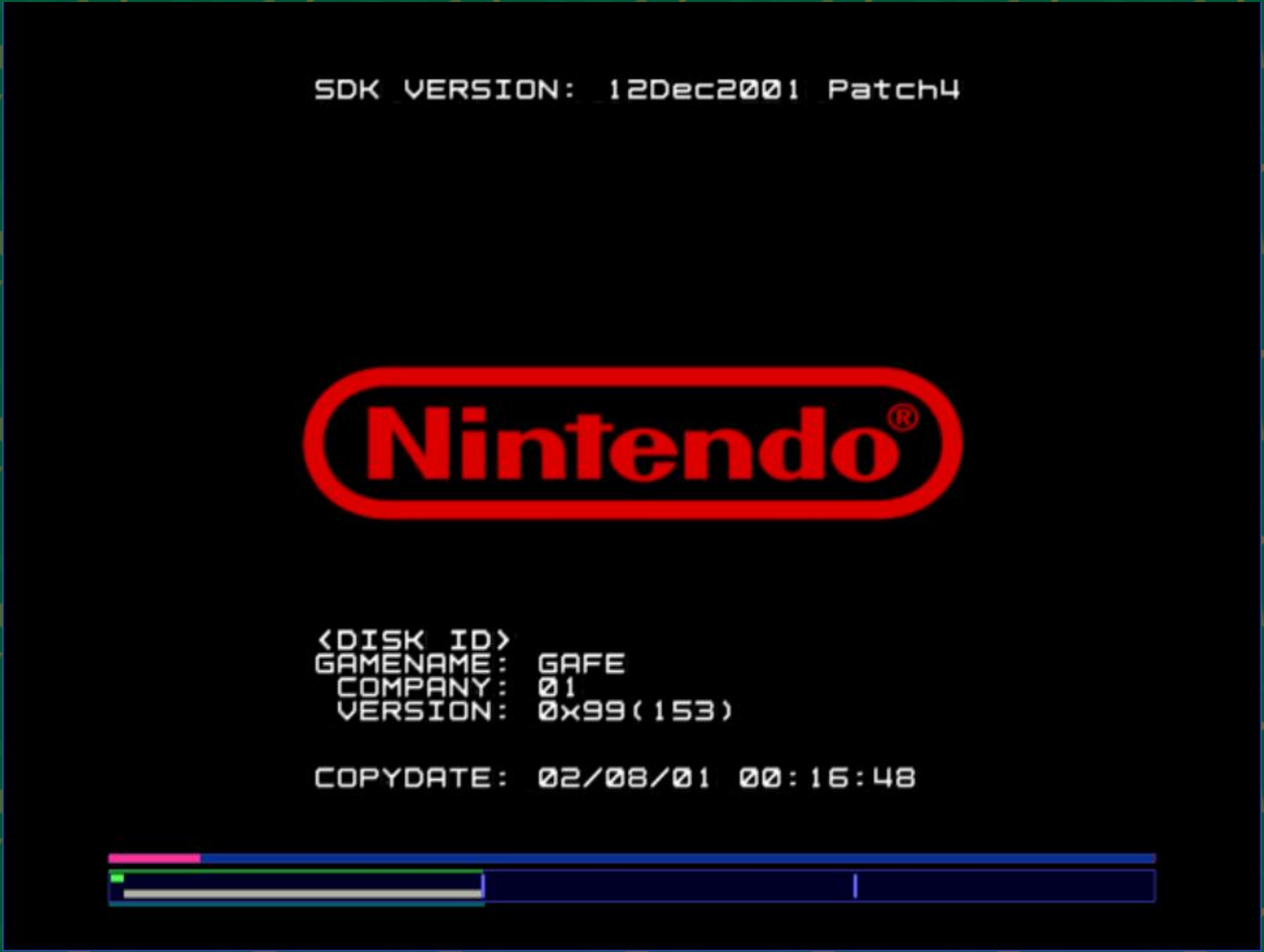


```
loc_800062C4:               # r3 = 0x80000000
bl          DVDGetCurrentDiskID
lbz         r0, 7(r3)       # Load Byte and Zero
cmplwi      r0, 0x99        # Compare Logical Word Immediate
bne         loc_800062E8    # Branch if not equal
```

```
lis         r4, osAppNMIBuffer@h # Load Immediate Shifted
addi        r4, r4, osAppNMIBuffer@l # Add Immediate
lwz         r0, (dword_80206F9C - osAppNMIBuffer)(r4) # Load Word and Zero
ori         r0, r0, 0x20  # Set bit 26 if version is 0x99 (enable zuru modes)
stw         r0, (dword_80206F9C - osAppNMIBuffer)(r4) # Store Word
```



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   47 41 46 45 30 31 00 00 00 00 00 00 00 00 00 00  GAFE01..........
00000010   00 00 00 00 00 00 00 00 00 00 00 00 C2 33 9F 3D  ............Â3Ÿ=
```



```
SDK VERSION: 12Dec2001 Patch4

<DISK ID>
GAMENAME: GAFE
 COMPANY: 01
 VERSION: 0x99(153)

COPYDATE: 02/08/01 00:16:48
```

Zuru Mode Activation

`0x3C(osAppNMIBuffer)`
Bit 26

Game disk ID is 0x99

Instant unlock

`0x3C(osAppNMIBuffer)`
Bit 25

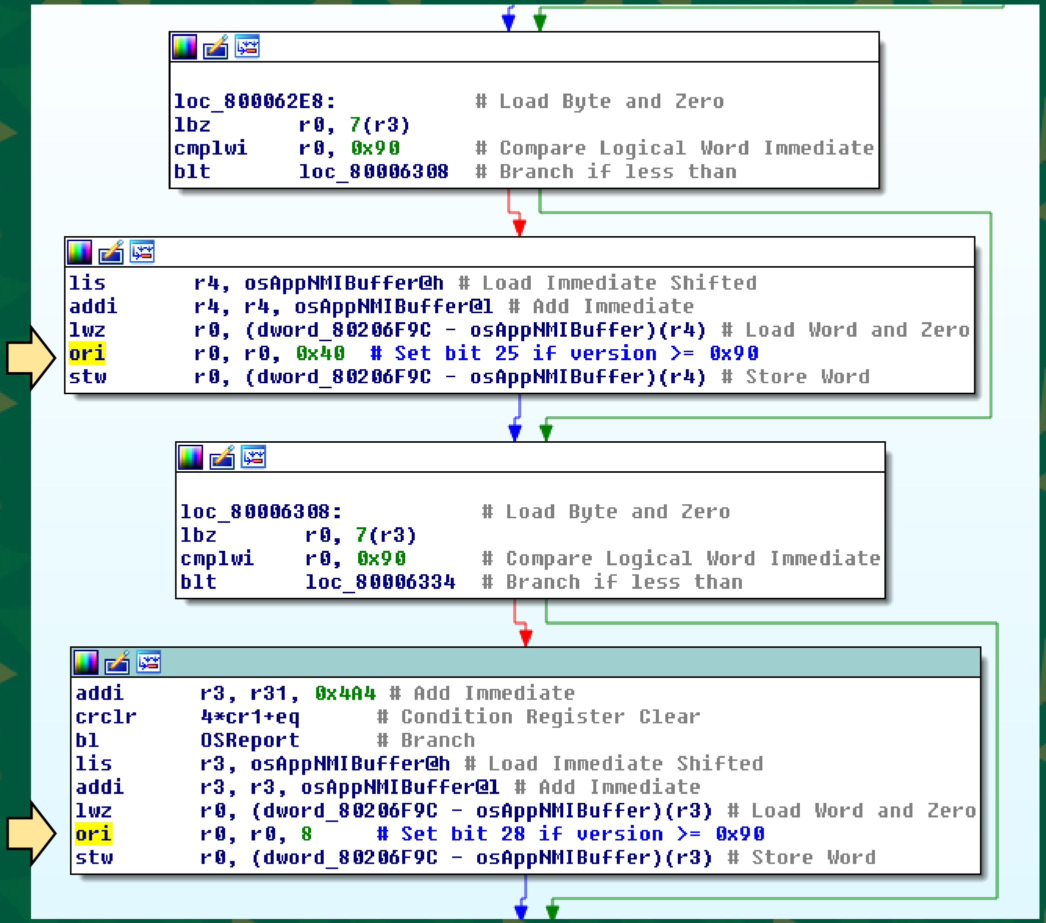
?

`0x4(zuruKeyCheck)`

?

Bit 25 and 28

- Bits 25 and 28 get set if the disk ID is greater than 0x90
- Bit 28 controls zuru mode level (1 or 2)
- Bit 25 was associated with that controller connection check…



```
loc_800062E8:                # Load Byte and Zero
lbz         r0, 7(r3)
cmplwi      r0, 0x90         # Compare Logical Word Immediate
blt         loc_80006308     # Branch if less than
```

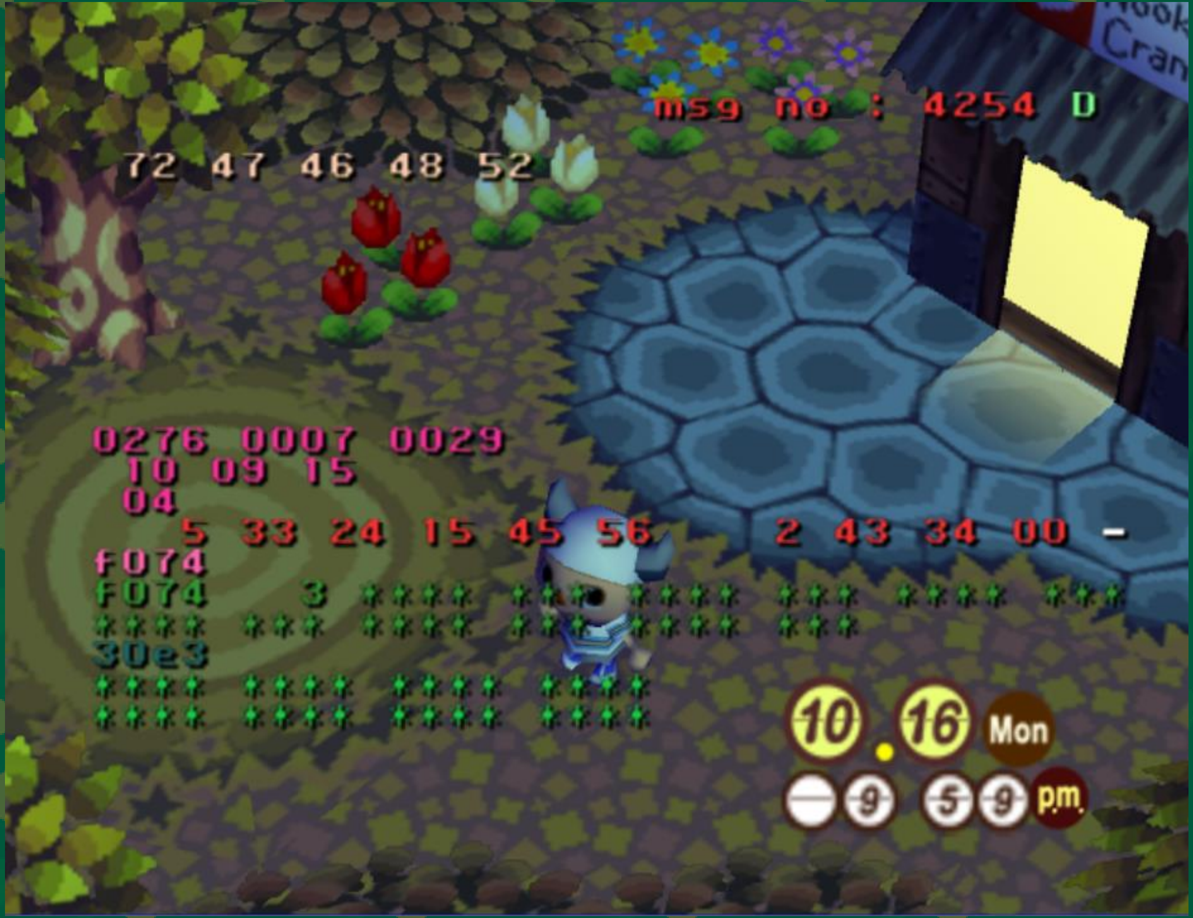
```
lis         r4, osAppNMIBuffer@h # Load Immediate Shifted
addi        r4, r4, osAppNMIBuffer@l # Add Immediate
lwz         r0, (dword_80206F9C - osAppNMIBuffer)(r4) # Load Word and Zero
ori         r0, r0, 0x40  # Set bit 25 if version >= 0x90
stw         r0, (dword_80206F9C - osAppNMIBuffer)(r4) # Store Word
```

```
loc_80006308:                # Load Byte and Zero
lbz         r0, 7(r3)
cmplwi      r0, 0x90         # Compare Logical Word Immediate
blt         loc_80006334     # Branch if less than
```

```
addi        r3, r31, 0x4A4 # Add Immediate
crclr       4*cr1+eq       # Condition Register Clear
bl          OSReport       # Branch
lis         r3, osAppNMIBuffer@h # Load Immediate Shifted
addi        r3, r3, osAppNMIBuffer@l # Add Immediate
lwz         r0, (dword_80206F9C - osAppNMIBuffer)(r3) # Load Word and Zero
ori         r0, r0, 8      # Set bit 28 if version >= 0x90
stw         r0, (dword_80206F9C - osAppNMIBuffer)(r3) # Store Word
```

Bit 25

- One of the conditions for enabling zuru mode was:
  - Bit 25 is set
  - A controller is connected to port 2
- If the game disk ID is between 0x90 and 0x98, zuru mode can be enabled by plugging in a second controller, and…
- The second controller controls all of the debug displays!



```
                                msg no : 4254 D
  72 47 46 48 52

0276 0007 0029
 10 09 15
 04
    5 33 24 15 45 56      2 43 34 00 -
f074
F074     3 **** *** **** *** **** ***
**** *** **** *** **** ***
30e3
**** **** **** ****
**** **** **** ****
```

Zuru Mode Activation

`0x3C(osAppNMIBuffer)`
Bit 26

Game disk ID is 0x99

Instant unlock

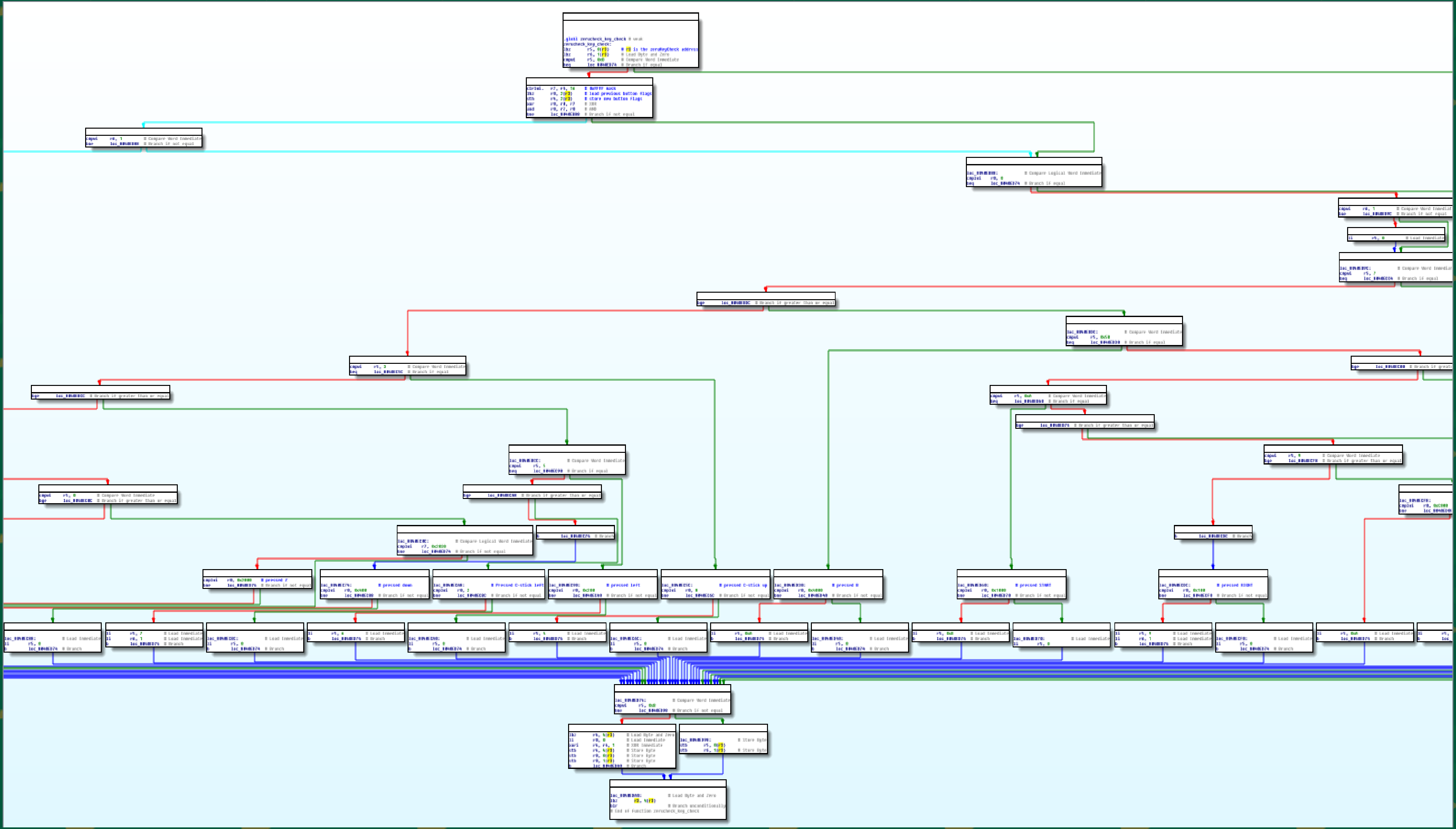
`0x3C(osAppNMIBuffer)`
Bit 25

Game disk ID between 0x90 and 0x98

Unlocked when a controller is in port 2

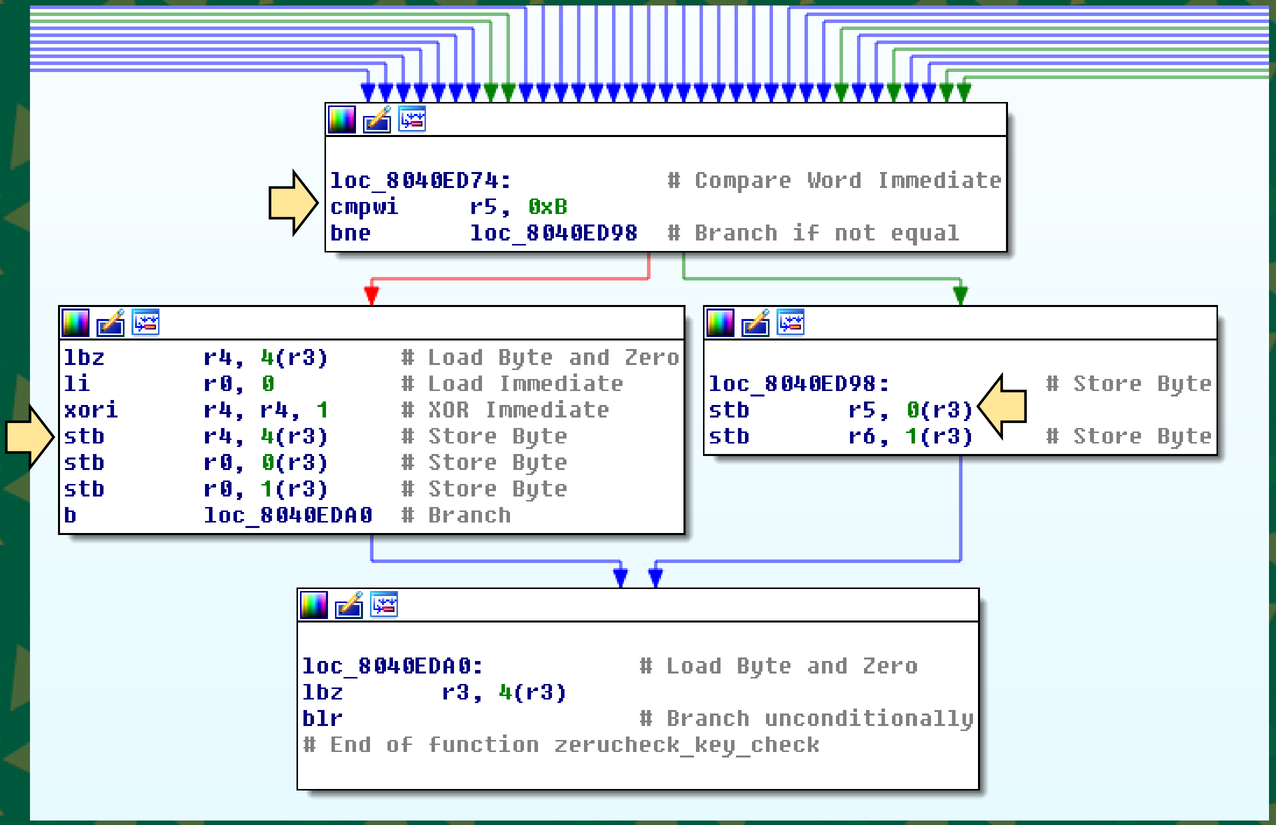
`0x4(zuruKeyCheck)`

?



zerucheck_key_check

- The last mystery is `zuruKeyCheck`
- It gets updated by `zerucheck_key_check`
  - Cross-reference didn't show up before because of the way the address is calculated
- What we want at the end is for register 5 to hold 0xB
  - This will toggle the value of 0x4(zuruKeyCheck), enabling or disabling zuru mode
- r5 is stored in `0x0(zuruKeyCheck)`
  - Loaded at the beginning
  - Updated at the end



```
loc_8040ED74:                 # Compare Word Immediate
cmpwi       r5, 0xB
bne         loc_8040ED98      # Branch if not equal
```

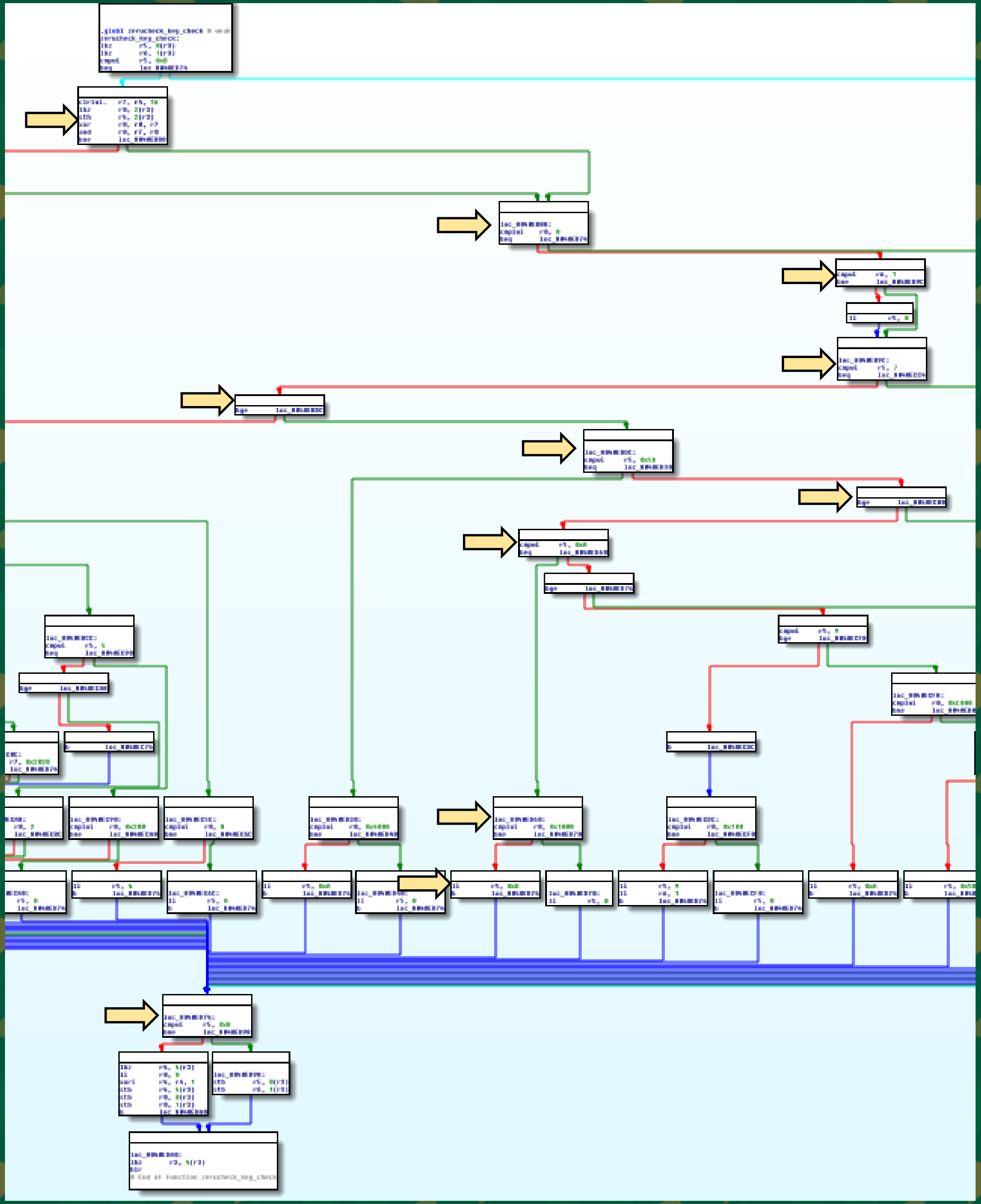
```
lbz         r4, 4(r3)      # Load Byte and Zero
li          r0, 0          # Load Immediate
xori        r4, r4, 1      # XOR Immediate
stb         r4, 4(r3)      # Store Byte
stb         r0, 0(r3)      # Store Byte
stb         r0, 1(r3)      # Store Byte
b           loc_8040EDA0   # Branch
```

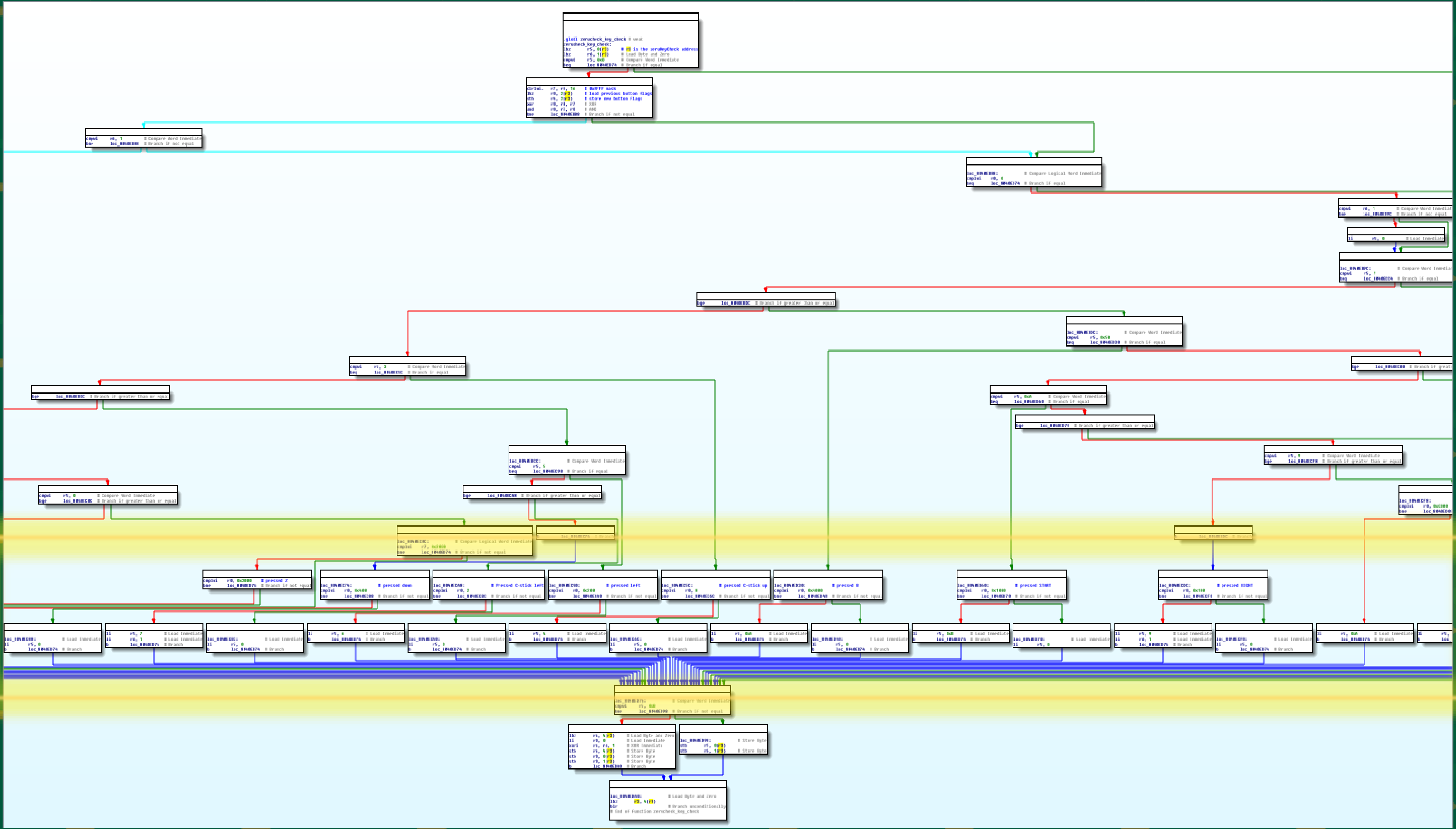
```
loc_8040ED98:                 # Store Byte
stb         r5, 0(r3)
stb         r6, 1(r3)         # Store Byte
```

```
loc_8040EDA0:                # Load Byte and Zero
lbz         r3, 4(r3)
blr                          # Branch unconditionally
# End of function zerucheck_key_check
```

zerucheck_key_check

- Follow the blocks up to the beginning and find the constraints
  - 8040ED74: r5 must be 0xB
  - Sets r5 to 0xB
  - 8040ED60: r0 must be 0x1000
  - 8040EBE8: r5 must be 0xA
  - 8040EBE4: r5 must be less than 0x5B
  - 8040EBA4: r5 must be greater than 0x7
  - 8040EB94: r6 must be 0x1
  - 8040EB5C: r0 must not be 0x0





```
.globl zerocheck_key_check # weak
zerocheck_key_check:
lbz       r5, 0(r3)      # r3 is the zeroKeyCheck address
lbz       r6, 1(r3)      # Load Byte and Zero
cmpwi     r5, 0x0        # Compare Word Immediate
beq       loc_[illegible] # Branch if equal
```

```
clrlwi.   r7, r4, 16     # BUTTON MASK
lbz       r8, 2(r3)      # load previous button flags
sth       r4, 2(r3)      # store new button flags
xor       r8, r8, r7     # XOR
and       r8, r7, r8     # AND
bne       loc_[illegible] # Branch if not equal
```

```
# pressed Z
# pressed down
# Pressed C-stick left
# pressed left
# Pressed C-stick up
# pressed B
# pressed START
# pressed RIGHT
```

```
loc_[illegible]:          # Compare Word Immediate
cmpwi     r5, 0x0
bne       loc_[illegible] # Branch if not equal
```

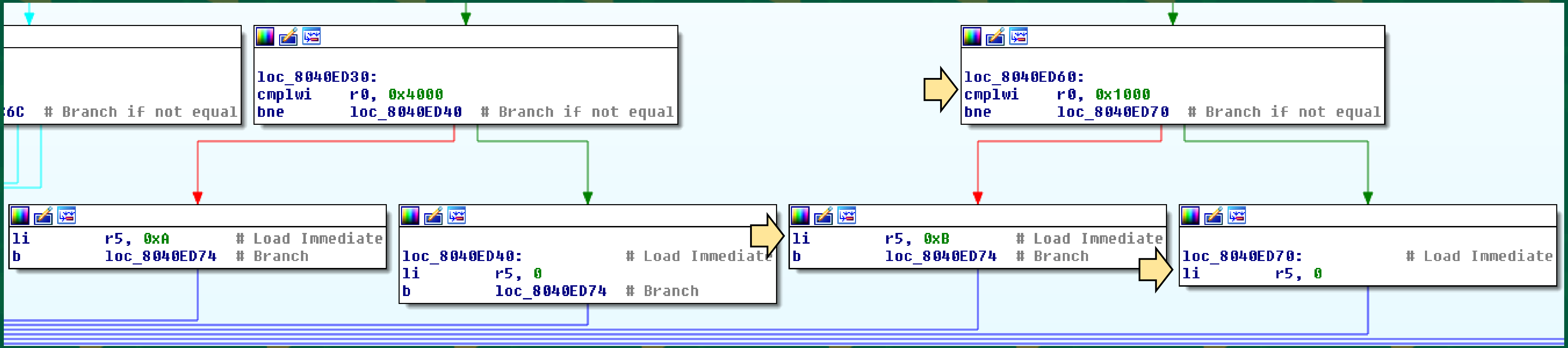
```
lbz       r4, 4(r3)      # Load Byte and Zero
li        r0, 0          # Load Immediate
xori      r4, r4, 1      # XOR Immediate
stb       r4, 4(r3)      # Store Byte
stb       r0, 0(r3)      # Store Byte
stb       r0, 1(r3)      # Store Byte
b         loc_[illegible] # Branch
```

```
loc_[illegible]:          # Store Byte
stb       r5, 0(r3)
stb       r6, 1(r3)      # Store Byte
```

```
loc_[illegible]:          # Load Byte and Zero
lbz       r3, 4(r3)
blr                      # Branch unconditionally
# End of function zerocheck_key_check
```

zerucheck_key_check

- The blocks right before the end will update r5 to some number or reset it to zero based on a comparison
- It’s a state machine
  - r5 stores state index and is advanced on correct conditions, or reset to zero
  - The condition is a comparison to the value of r0



```
6C    # Branch if not equal
```

```
loc_8040ED30:
cmplwi    r0, 0x4000
bne       loc_8040ED40    # Branch if not equal
```

```
loc_8040ED60:
cmplwi    r0, 0x1000
bne       loc_8040ED70    # Branch if not equal
```

```
li          r5, 0xA            # Load Immediate
b           loc_8040ED74       # Branch
```

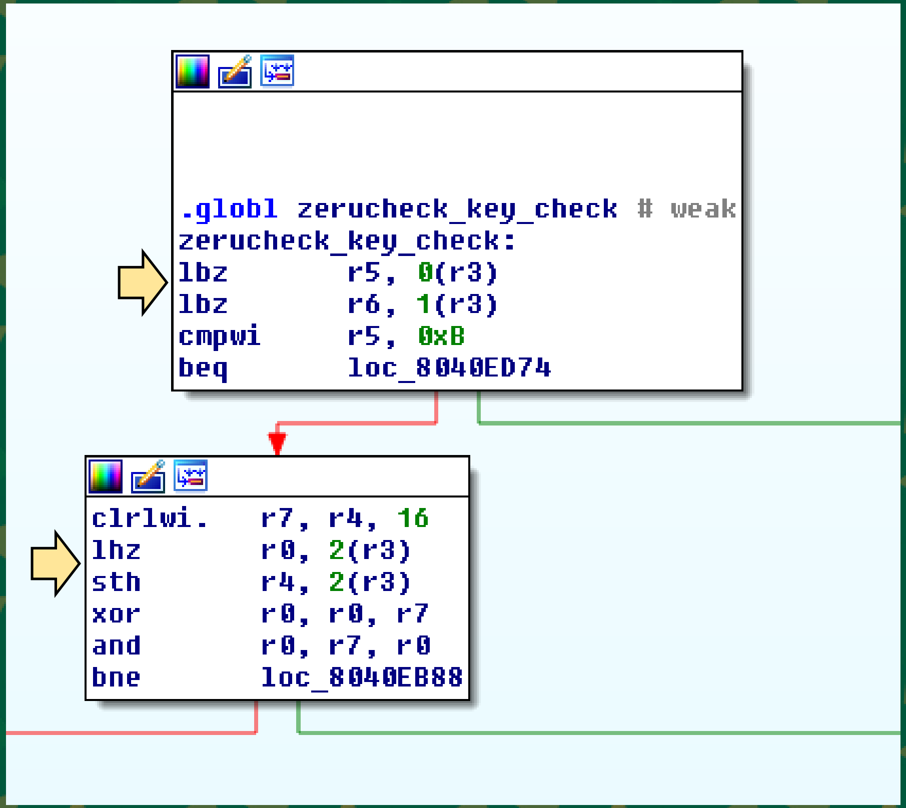
```
loc_8040ED40:                  # Load Immediate
li          r5, 0
b           loc_8040ED74       # Branch
```

```
li          r5, 0xB            # Load Immediate
b           loc_8040ED74       # Branch
```

```
loc_8040ED70:                  # Load Immediate
li          r5, 0
```

zerucheck_key_check

- The values of r0 looks like bit flags…
  - Where do they come from?
- Function called every frame via callback function passed to gamepad manager class
- Holding down various buttons on the second controller changes the value
  - Affects 16-bit value at offset `0x2`
  - So it *is* checking for certain button combinations on a controller
- The first thing key check does is load the state
- Second thing is load the previous and current button press flags
  - `(new XOR old) AND new` leaves only the changed button press flags
  - The input to this function is new button presses – this is r0



```
.globl zerucheck_key_check # weak
zerucheck_key_check:
lbz         r5, 0(r3)
lbz         r6, 1(r3)
cmpwi       r5, 0xB
beq         loc_8040ED74
```

```
clrlwi.     r7, r4, 16
lhz         r0, 2(r3)
sth         r4, 2(r3)
xor         r0, r0, r7
and         r0, r7, r0
bne         loc_8040EB88
```

zerucheck_key_check

- Look up button values in N64 SDK
- It’s a cheat combo!
    1. Hold L + R triggers and press Z
    2. D-UP
    3. C-DOWN
    4. C-UP
    5. D-DOWN
    6. D-LEFT
    7. C-LEFT
    8. C-RIGHT
    9. D-RIGHT
    10. A + B
    11. START

Button	Value
A_BUTTON	0x8000
B_BUTTON	0x4000
L_TRIG	0x0020
R_TRIG	0x0010
Z_TRIG	0x2000
START_BUTTON	0x1000
U_JPAD	0x0800
L_JPAD	0x0200
R_JPAD	0x0100
D_JPAD	0x0400
U_CBUTTONS	0x0008
L_CBUTTONS	0x0002
R_CBUTTONS	0x0001
D_CBUTTONS	0x0004

Zuru Mode Activation

`0x3C(osAppNMIBuffer)`
Bit 26

Game disk ID is 0x99

Instant unlock

`0x3C(osAppNMIBuffer)`
Bit 25

Game disk ID between 0x90 and 0x98

Unlocked when a controller is in port 2

`0x4(zuruKeyCheck)`

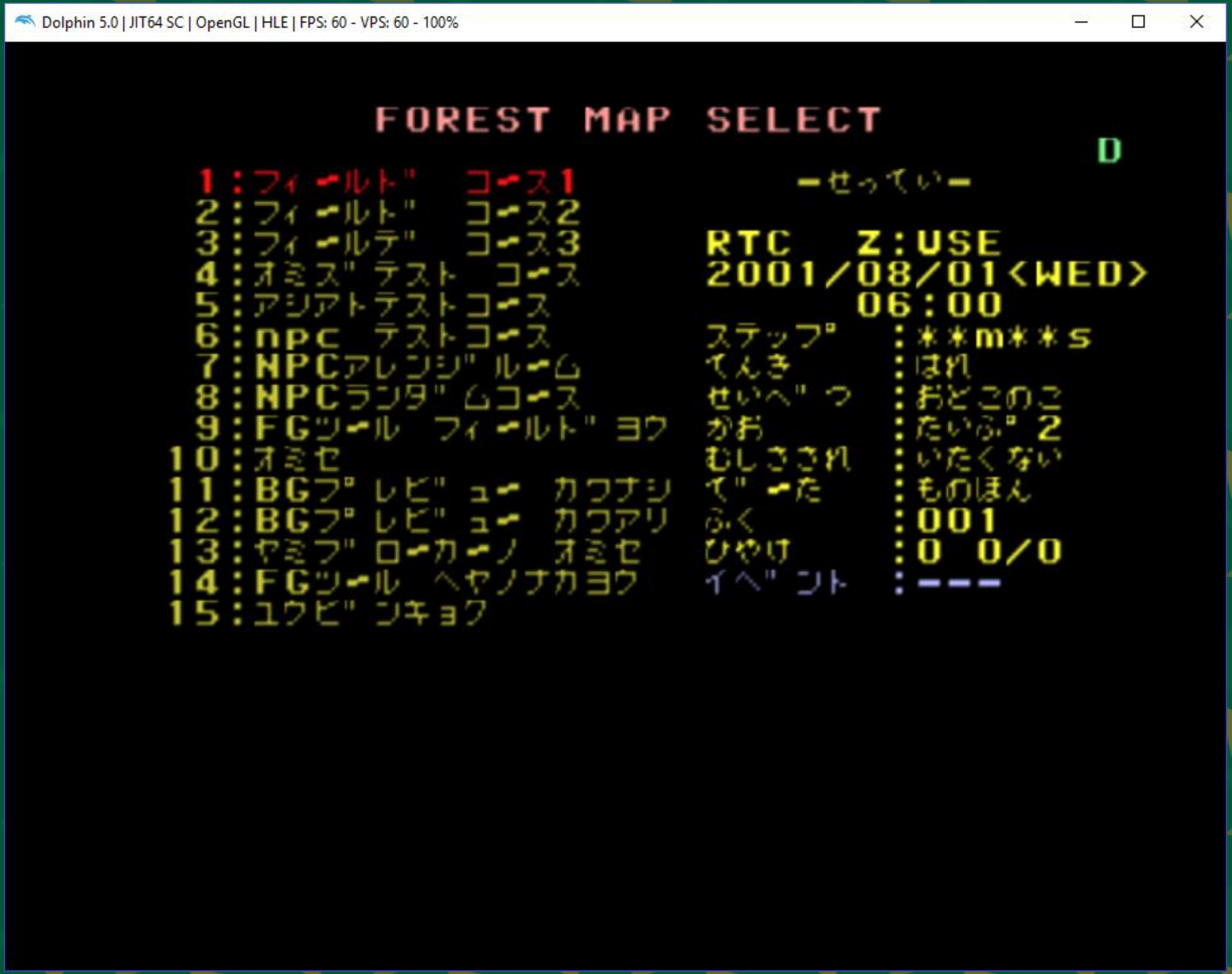
Enter 11-step button combo on port 2 controller

Toggle unlock with button combo

2017 11/13 13:56 49
53 55
5891
38 0 - 6 - 0
7 5 -
2 2
4 255
76 13-15-16
0307 0000 1301
00 00 3D
00
0 00 00 00 00 00 0 00 00 00
3098
3094
11 13 Mon
1:56 p.m.
VIZIO

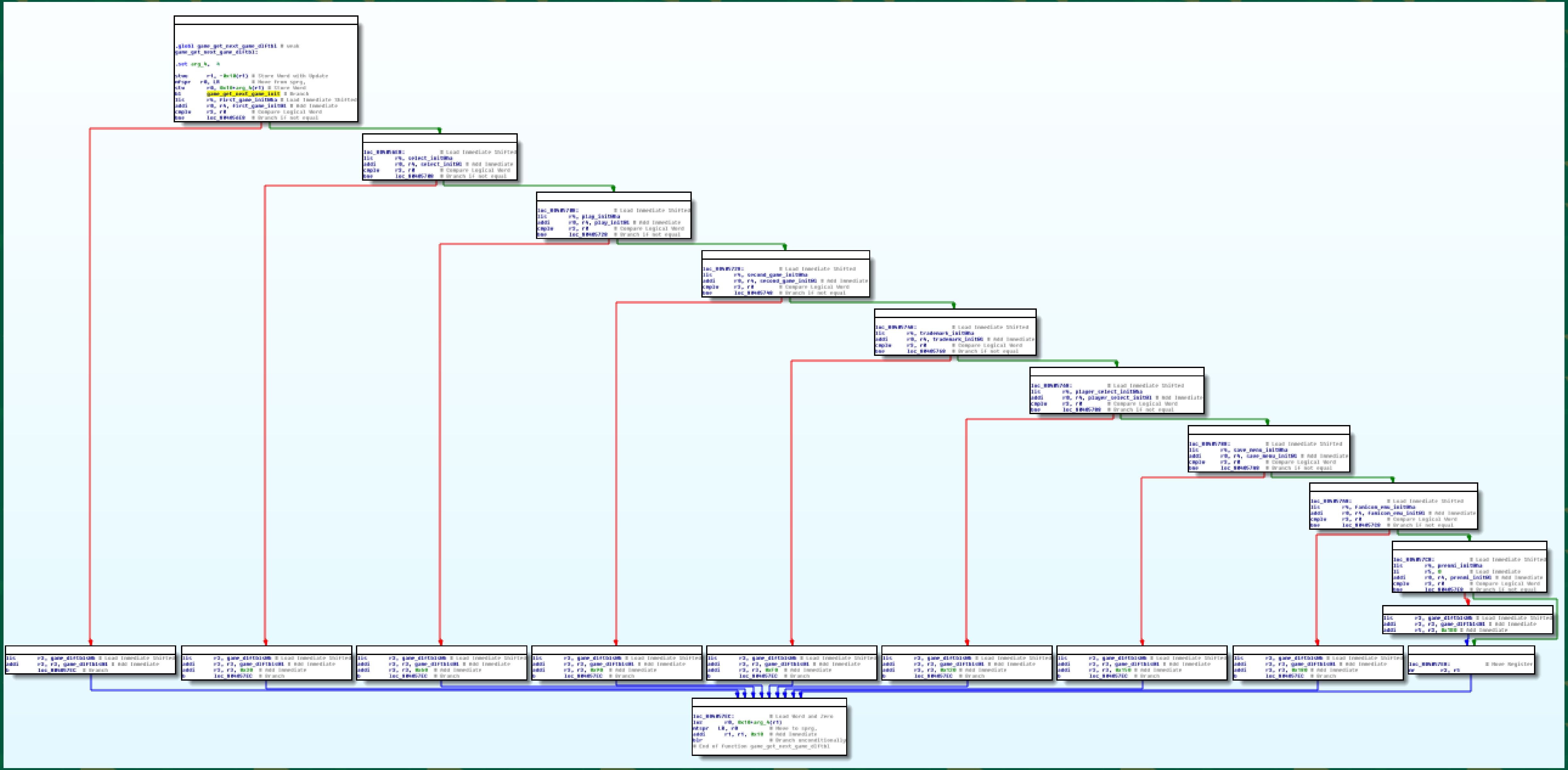
Special menus

- Famicom menu
- Map select
- Player select
- Scene selection



Dolphin 5.0 | JIT64 SC | OpenGL | HLE | FPS: 60 - VPS: 60 - 100%

```
                FOREST MAP SELECT
                                                    D
    1:フィールド  コース1         ーせってぃー
    2:フィールド  コース2
    3:フィールデ"  コース3     RTC   Z:USE
    4:オミズ テスト  コース     2001/08/01<WED>
    5:アシアトテストコース             06:00
    6:npc テストコース        ステップ    :**m**s
    7:NPCアレンジルーム        てんき      :はれ
    8:NPCランダムコース        せいべつ    :おとこのこ
    9:FGツール フィールドヨウ   かお        :たいぷ2
   10:オミセ                 むしさされ  :いたくない
   11:BGプレビュー カワナシ    でーた      :ものほん
   12:BGプレビュー カワアリ    ふく        :001
   13:ヤミブローカーノ オミセ   ひやけ      :0 0/0
   14:FGツール ヘヤノナカヨウ   イベント    :---
   15:ユウビンキョク
```



```
.globl game_get_next_game_dlftbl # weak
game_get_next_game_dlftbl:

.set arg_4,  4

stwu      r1, -0x10(r1) # Store Word with Update
mflr    r0, LR          # Move from sprg,
stw       r0, 0x10+arg_4(r1) # Store Word
bl        game_get_next_game_init # Branch
lis       r4, first_game_init@ha # Load Immediate Shifted
addi      r0, r4, first_game_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, select_init@ha
addi      r0, r4, select_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, play_init@ha
addi      r0, r4, play_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, second_game_init@ha
addi      r0, r4, second_game_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, trademark_init@ha
addi      r0, r4, trademark_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, player_select_init@ha
addi      r0, r4, player_select_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, save_menu_init@ha
addi      r0, r4, save_menu_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, famicom_emu_init@ha
addi      r0, r4, famicom_emu_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
loc_8040[illegible]:             # Load Immediate Shifted
lis       r4, prenmi_init@ha
li        r5, 0          # Load Immediate
addi      r0, r4, prenmi_init@l # Add Immediate
cmplw     r3, r0        # Compare Logical Word
bne       loc_8040[illegible] # Branch if not equal
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r5, r3, 0x1B0 # Add Immediate
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x30  # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x60  # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x90  # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0xC0  # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0xF0  # Add Immediate
b         loc_804057EC  # Branch
```

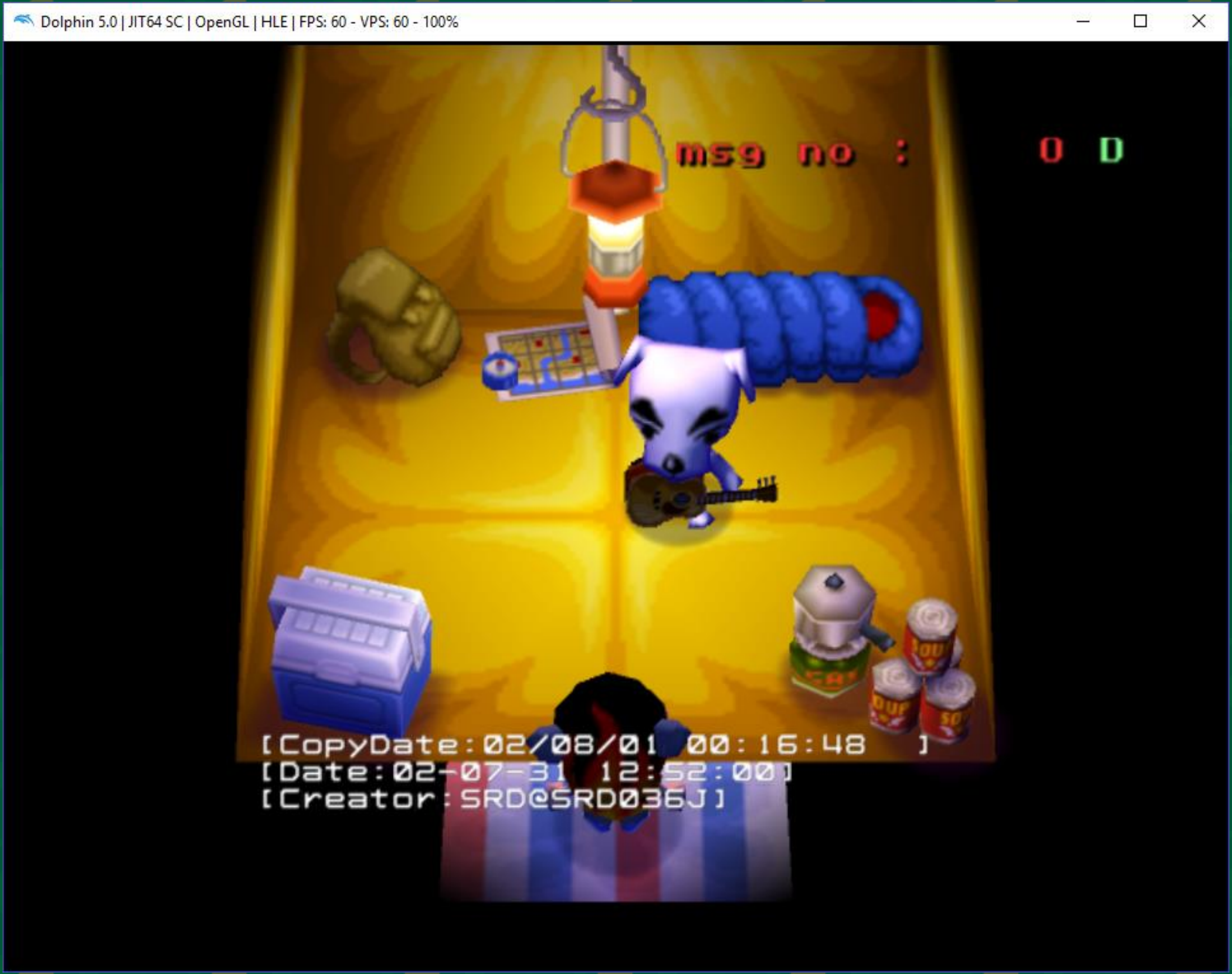
```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x120 # Add Immediate
b         loc_804057EC  # Branch
```

```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x150 # Add Immediate
b         loc_804057EC  # Branch
```

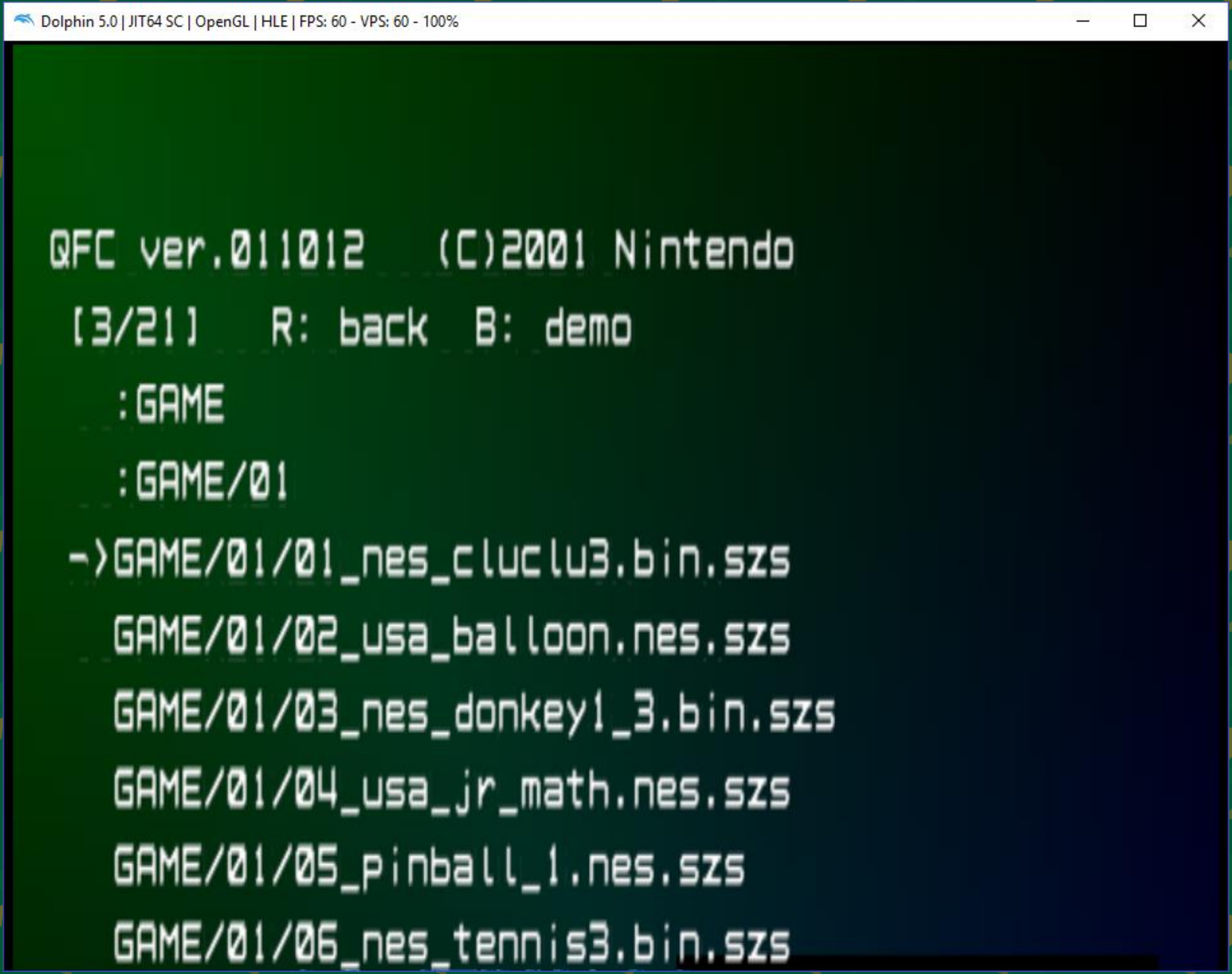
```
lis       r3, game_dlftbl@h # Load Immediate Shifted
addi      r3, r3, game_dlftbl@l # Add Immediate
addi      r3, r3, 0x180 # Add Immediate
b         loc_804057EC  # Branch
```

```
loc_804057E8:                    # Move Register
mr        r3, r5
```

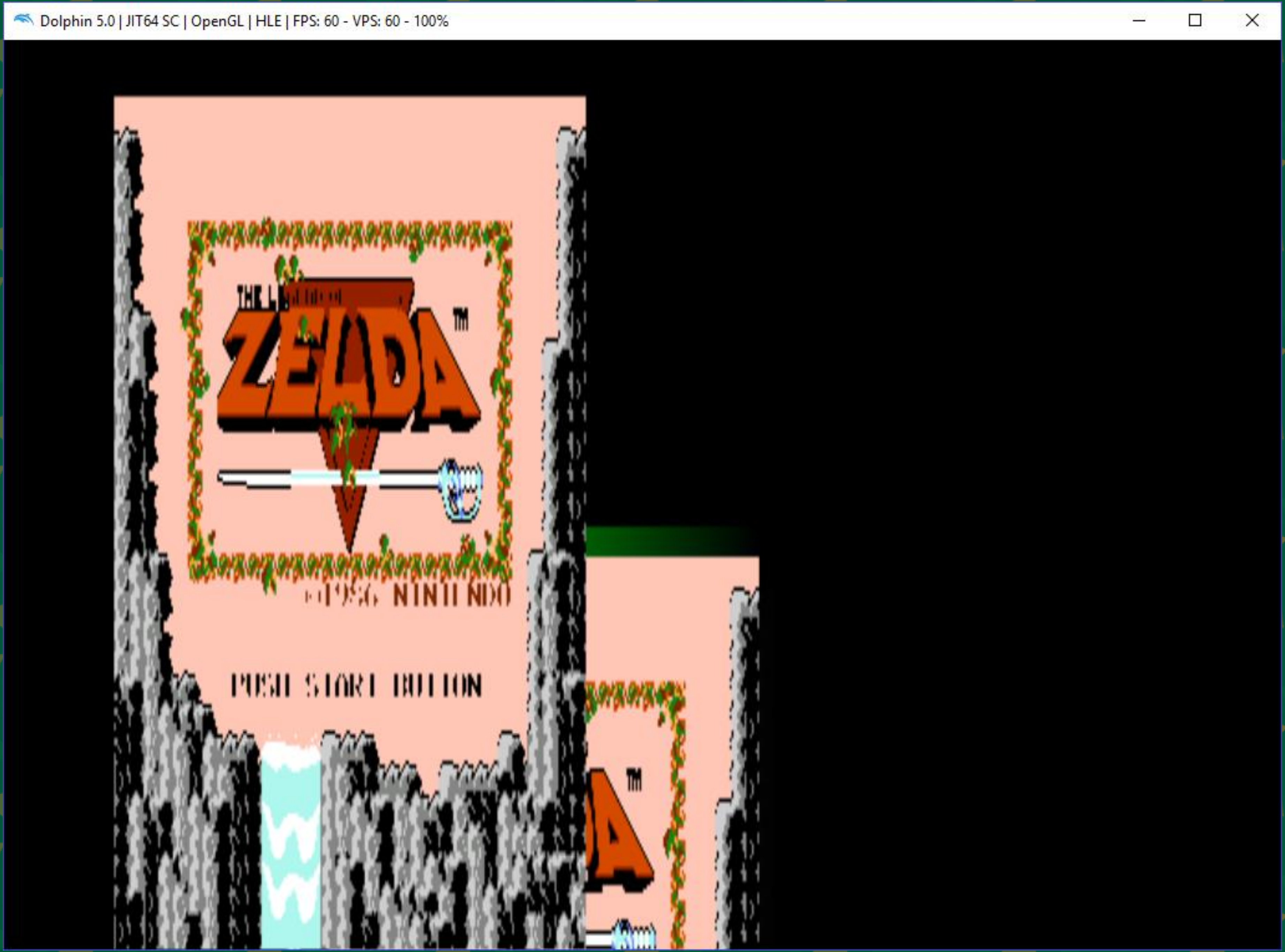
```
loc_804057EC:                    # Load Word and Zero
lwz       r0, 0x10+arg_4(r1)
mtspr   LR, r0                  # Move to sprg,
addi      r1, r1, 0x10  # Add Immediate
blr                             # Branch unconditionally
# End of function game_get_next_game_dlftbl
```


Dolphin 5.0 | JIT64 SC | OpenGL | HLE | FPS: 60 - VPS: 60 - 100%

```
msg no :      0  D
[CopyDate:02/08/01 00:16:48   ]
[Date:02-07-31 12:52:00]
[Creator:SRD@SRD036J]
```


Dolphin 5.0 | JIT64 SC | OpenGL | HLE | FPS: 60 - VPS: 60 - 100%

```
QFC ver.011012   (C)2001 Nintendo
 [3/21]   R: back  B: demo
   :GAME
   :GAME/01
 ->GAME/01/01_nes_cluclu3.bin.szs
   GAME/01/02_usa_balloon.nes.szs
   GAME/01/03_nes_donkey1_3.bin.szs
   GAME/01/04_usa_jr_math.nes.szs
   GAME/01/05_pinball_1.nes.szs
   GAME/01/06_nes_tennis3.bin.szs
```


Dolphin 5.0 | JIT64 SC | OpenGL | HLE | FPS: 60 - VPS: 60 - 100%
ZELDA
TM
©1986 NINTENDO
PUSH START BUTTON

Bonus

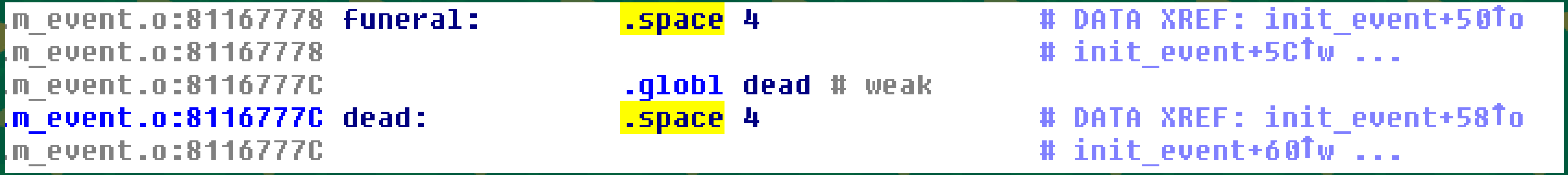
Translations, localization, development history

Haniwa / Gyroids

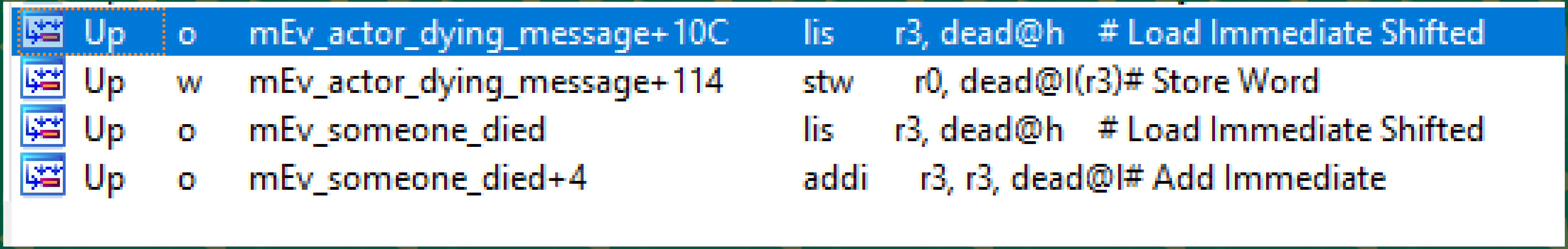
Kamakura

- Googling it returns a city
- Look up related message ID in message table:
  - “So what do you think? Isn't this a great igloo, {{TAIL}}?”
- Originally based on snow hut festival in Japan
  - Igloos are the localized version

Death / Funeral



```
.m_event.o:81167778 funeral:        .space 4                # DATA XREF: init_event+50↑o
.m_event.o:81167778                                         # init_event+5C↑w ...
.m_event.o:8116777C                 .globl dead # weak
.m_event.o:8116777C dead:           .space 4                # DATA XREF: init_event+58↑o
.m_event.o:8116777C                                         # init_event+60↑w ...
```



```
Up  o  mEv_actor_dying_message+10C  lis   r3, dead@h    # Load Immediate Shifted
Up  w  mEv_actor_dying_message+114  stw   r0, dead@l(r3)# Store Word
Up  o  mEv_someone_died             lis   r3, dead@h    # Load Immediate Shifted
Up  o  mEv_someone_died+4           addi  r3, r3, dead@l# Add Immediate
```