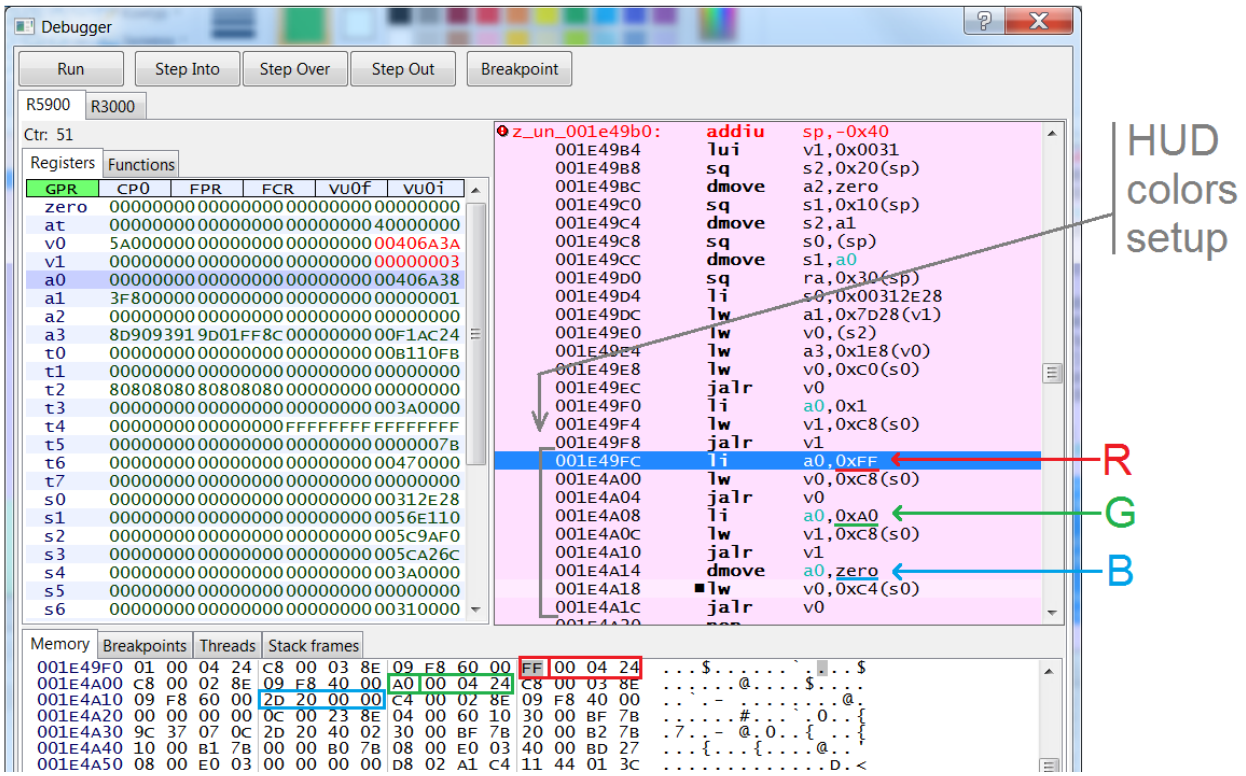
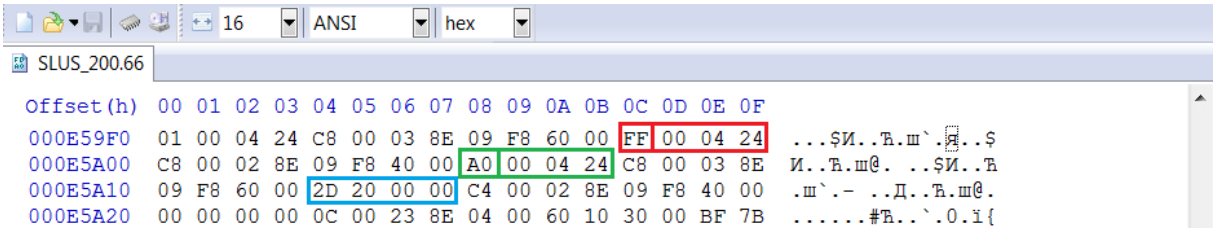
1 Changing color of the HUD

HUD color is baked into the executable file so the only way to change it is to do hacks. I was able to track down the point where the HUD color is set in PCSX2 debugger:

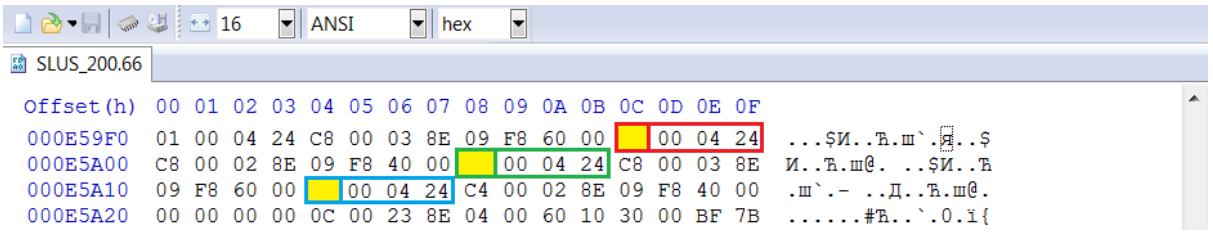


As you can see, changing red and green components is as simple as overriding operands. However, because blue component is equal to zero separate instruction for loading zero is used for it. So, to change blue component we need to change instruction to the one that is used for R and G.

Here is the same fragment, but inside an executable file (Offsets: 0xE59FC, 0xE5A08, 0xE5A14):



Here is how you change HUD colors:



First, you need to change last 3 bytes inside blue box from 0x20, 0x00, 0x00 to 0x00, 0x04, 0x24. Then all you need to do is just to input R, G and B components of desired color in yellow fields. That is it.

2 Enabling developer console

YouTube user GeckonCZ discovered that developer console is still present in PS2 Half-Life and can be accessed via a single byte hack. Link to his video:

https://www.youtube.com/watch?v=TZpNdbHtSw8&t=1s

These are known offsets that were published by him:

“HL PS2 USA ver 0.10: offset 0x00165428, change 01 to 00

HL PS2 USA ver 2.40: offset 0x00166988, change 01 to 00”

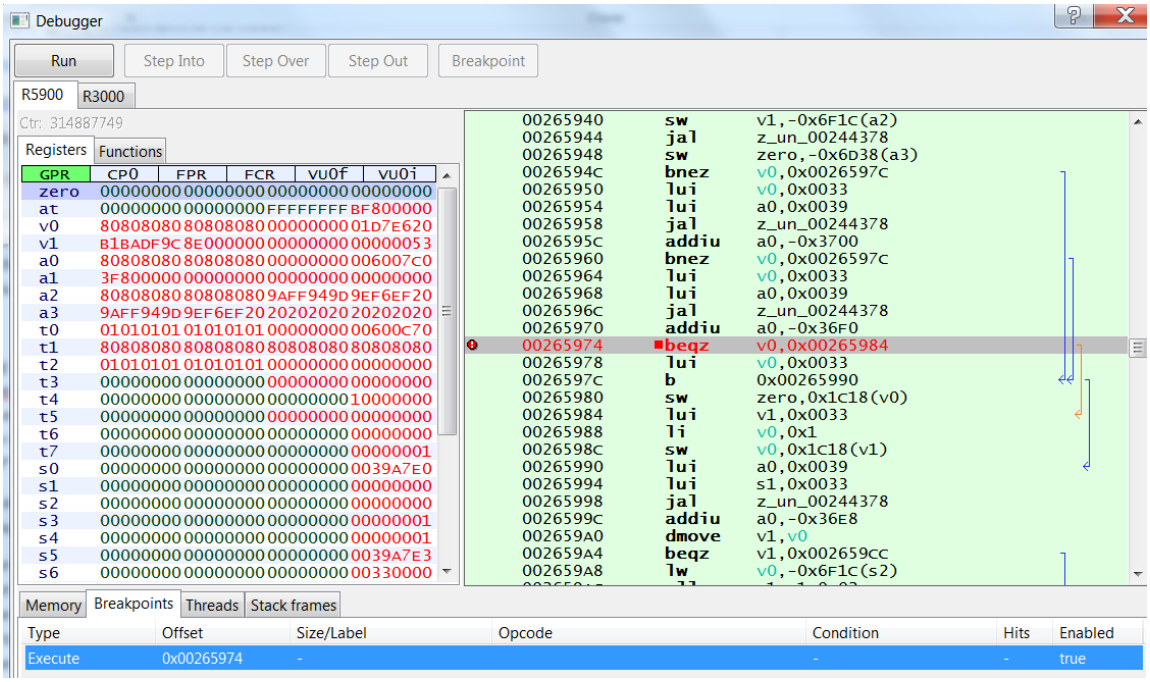
At the time I found this video he kept offsets in private. So I grinded my way to the console in PCSX2 debugger and I can add information on how this hack works in low level. Code that serves "toggleconsole" command checks for some flag and opens console only if this flag is set to 0. By default, this flag is set to 1 thus restricting console to show up. Here is a code that sets the flag:

```
00265974      beqz    v0,0x00265984
00265978      lui     v0,0x0033
0026597c      b       0x00265990
00265980      sw      zero,0x1c18(v0)
00265984      lui     v1,0x0033
00265988      li      v0,0x1
0026598c      sw      v0,0x1c18(v1)
00265990      lui     a0,0x0039
00265994      lui     s1,0x0033
00265998      jal     0x00244378
```

By changing this byte you effectively changing operand of selected instruction to 0 and thus setting the flag to 0. GeckonCZ later explained how it works on high level: “It just checks for the "-console" launch arguments as all the WON versions do. If that switch is found it sets the console CVAR to 1. And that's it... I'm not sure if you can easily pass launch arguments to PS2 executable so for me it was easier to patch the binary itself”.

What I can also add is the way to enable console in PCSX2 without changing files in hex editor using just built in debugger:

1) Start PCSX2.
2) Start PS2 Half-Life.
3) Open PCSX2 debugger (Debug->Open debug window …)
4) Go to Breakpoints tab and add new execute breakpoint at 0x265974.
5) Reset the game.
6) Debugger should pop up and show this:

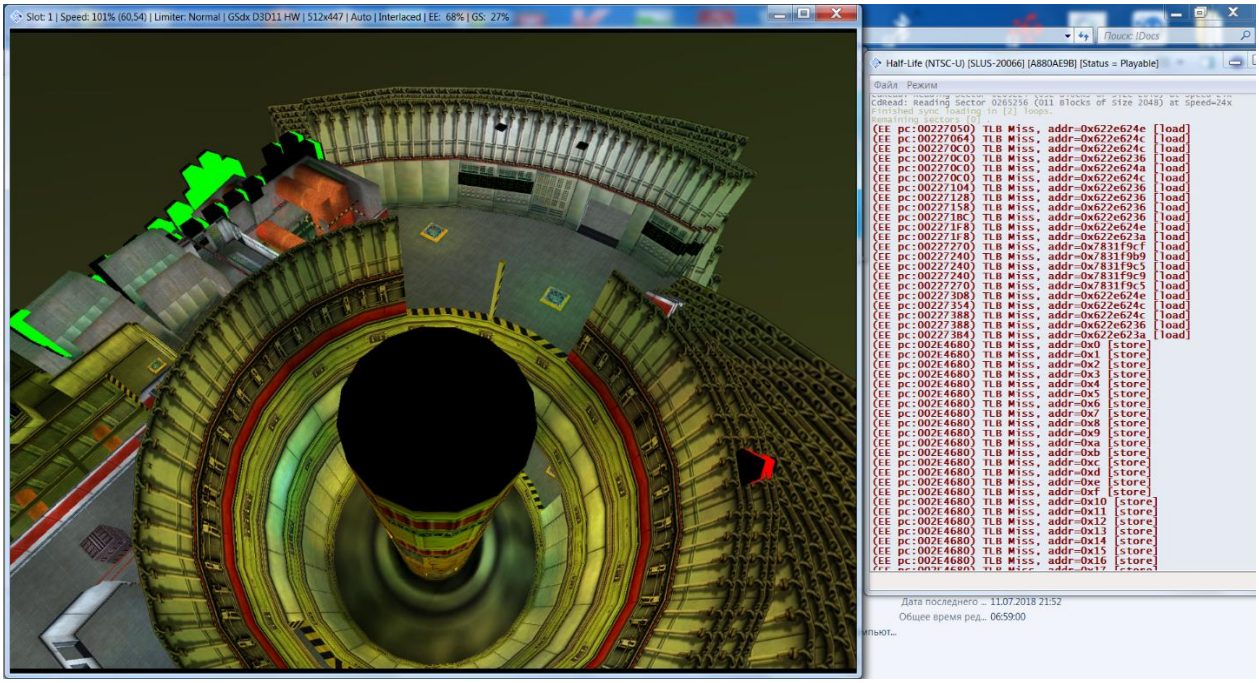


7) Press right mouse button on the next instruction after red line (lui v0,0x0033) and select “Jump to cursor” option.
8) Hit Run button. That is it, console should be enabled now.

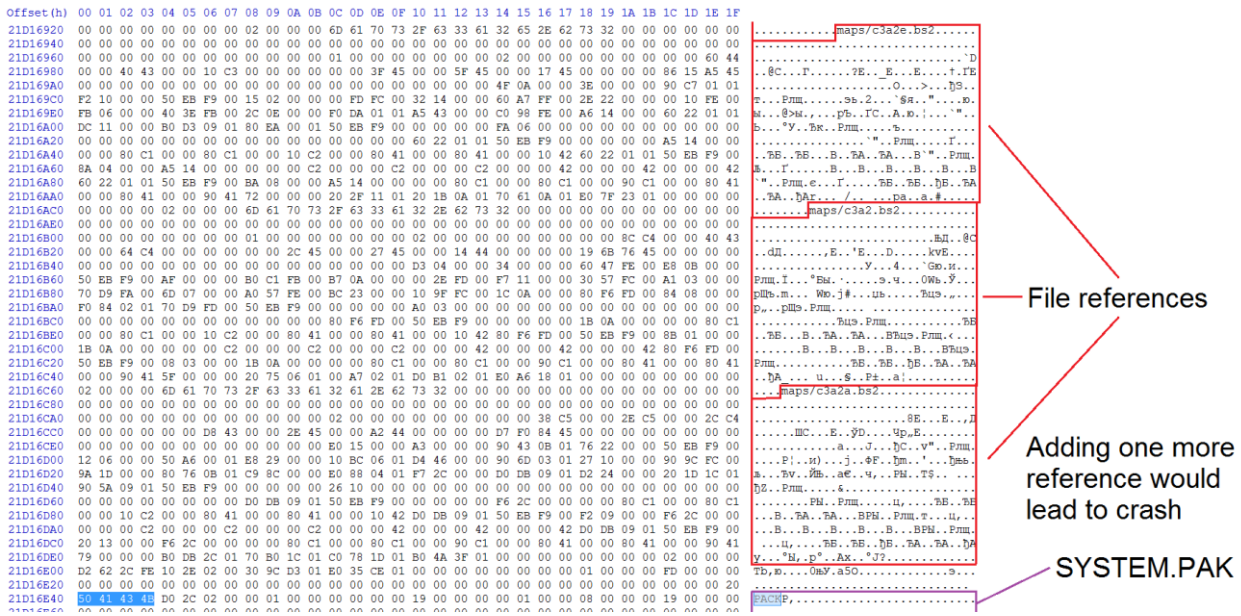
3 Improving stability by increasing file reference buffer size

There is section in the RAM where a reference to every used map, model and sprite file is stored. Handling of this section is broken: references to files that are no longer used are remaining forever and not cleaned, thus causing this section to grow indefinitely towards SYSTEM.PAK section with each new file used. When it reaches SYSTEM.PAK section, it can either cause spawning out of bounds or game crash. And the only way to clean out this thing is to hit reset button on PS2.

I had successfully recreated this bug on vanilla PS2 HL game: I played in one sitting through hazard course and campaign, and then in the “Lambda Reactor” chapter I got out of bounds spawn and crash after I went through Portal 7:



Here is what RAM looked like right before the crash:



This is quite good endurance for original game, but in my Blue Shift mod this bug caused crash on the beginning of “Focal point” chapter of one sitting playthrough.

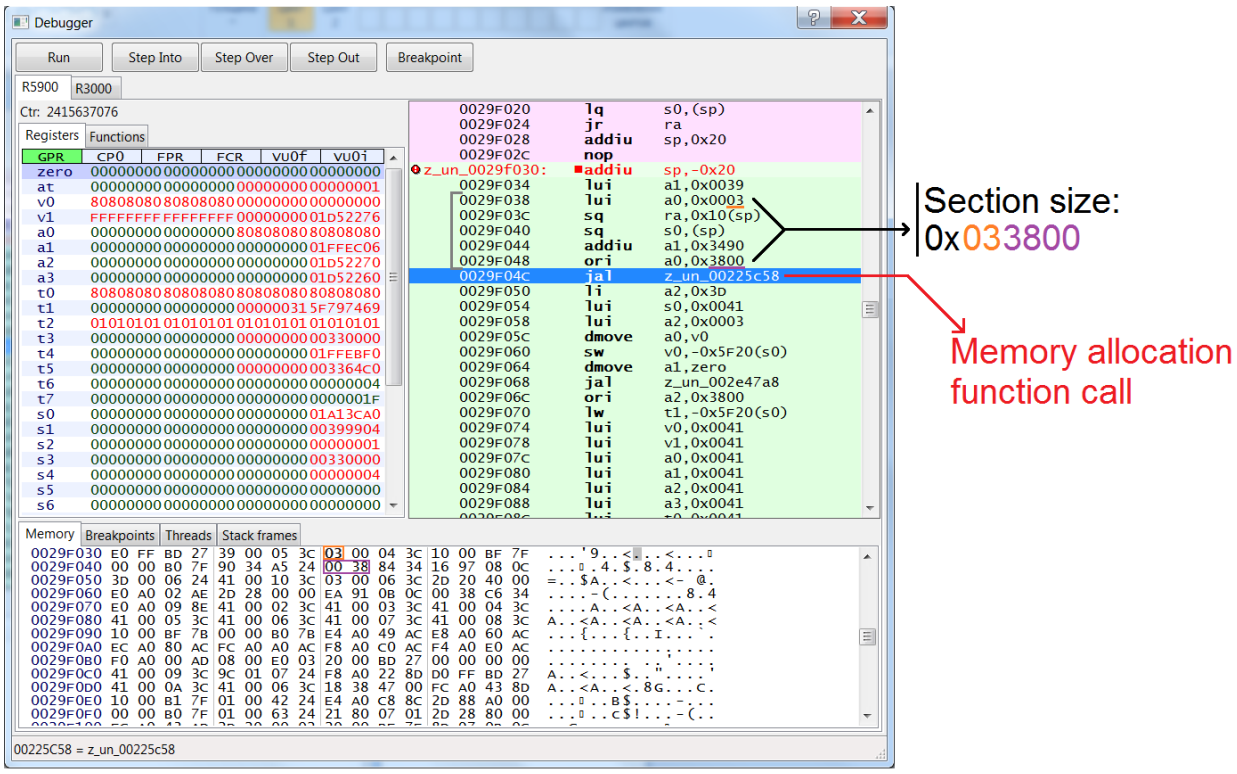
The good thing is that this section is allocated dynamically, which I determined by presence of this header that Is placed right before it:

???	Current section size	Ptr. to next section	Ptr. to prev. section
82 8E 2E FE	00 38 03 00	80 70 D1 01	00 E2 AE 00

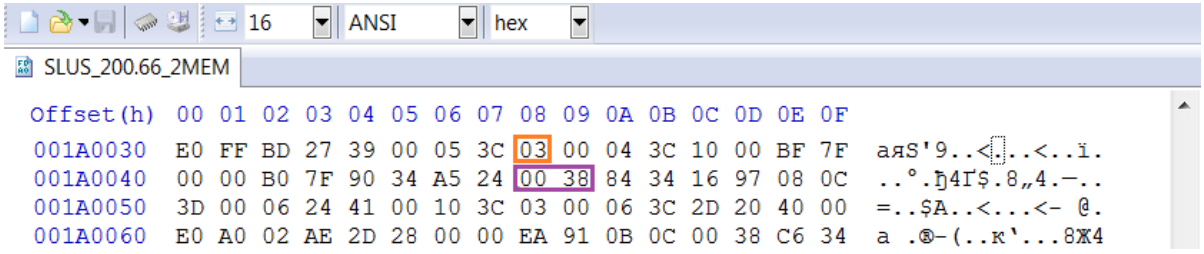
So a size of this section is 0x33800 bytes. Each file reference takes 0x19C bytes. So:

0x33800 / 0x19C = 0x200 = 512 – is maximum amount of files that can be used during one playthrough.

Because I had found exact location of section size field I used PCSX2 debugger to find place where this value came from:



Then I found location of these operands in executable file (offsets: 0x1A0038, 0x1A0048):



Then I rewrote them to allocate 0x80000 bytes (512 KiB) and It helped to stabilize Blue Shift so it was possible to finish it in one sitting: