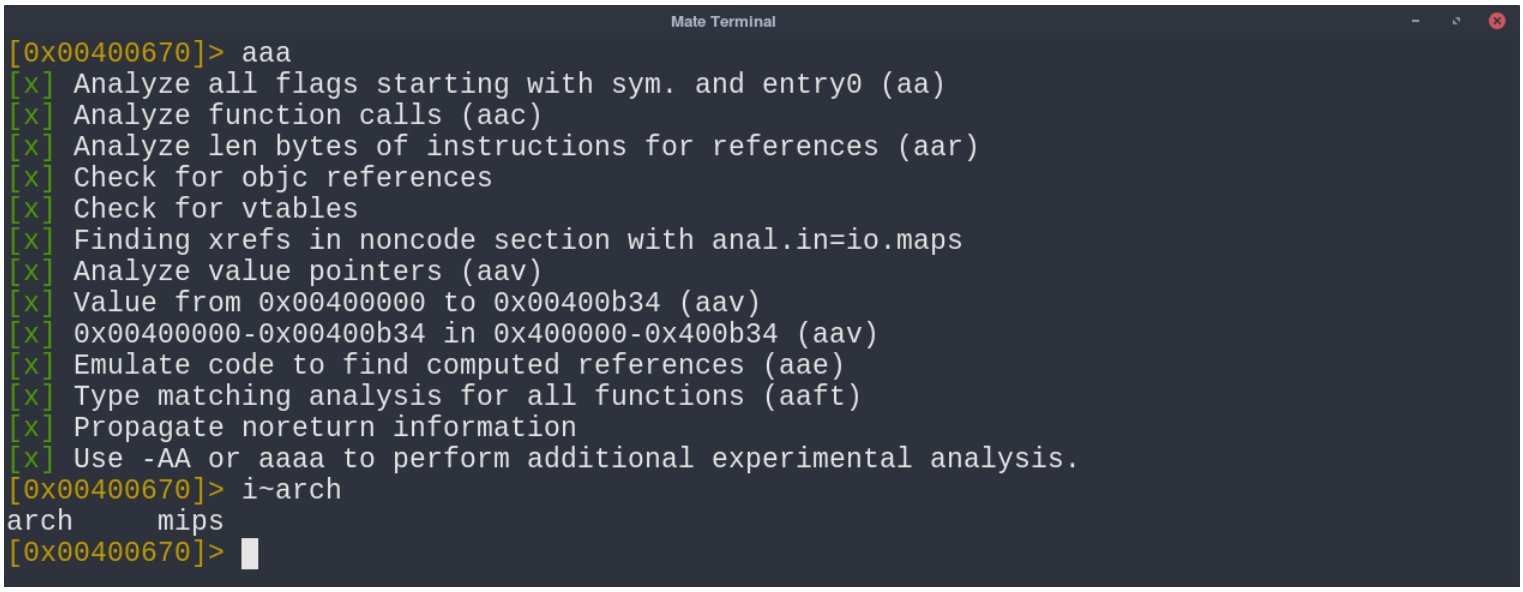
# Emulating a MIPS binary with radare2

This writeup shows how to use ESIL in radare2 to emulate a mips binary and solve a ctf challenge.

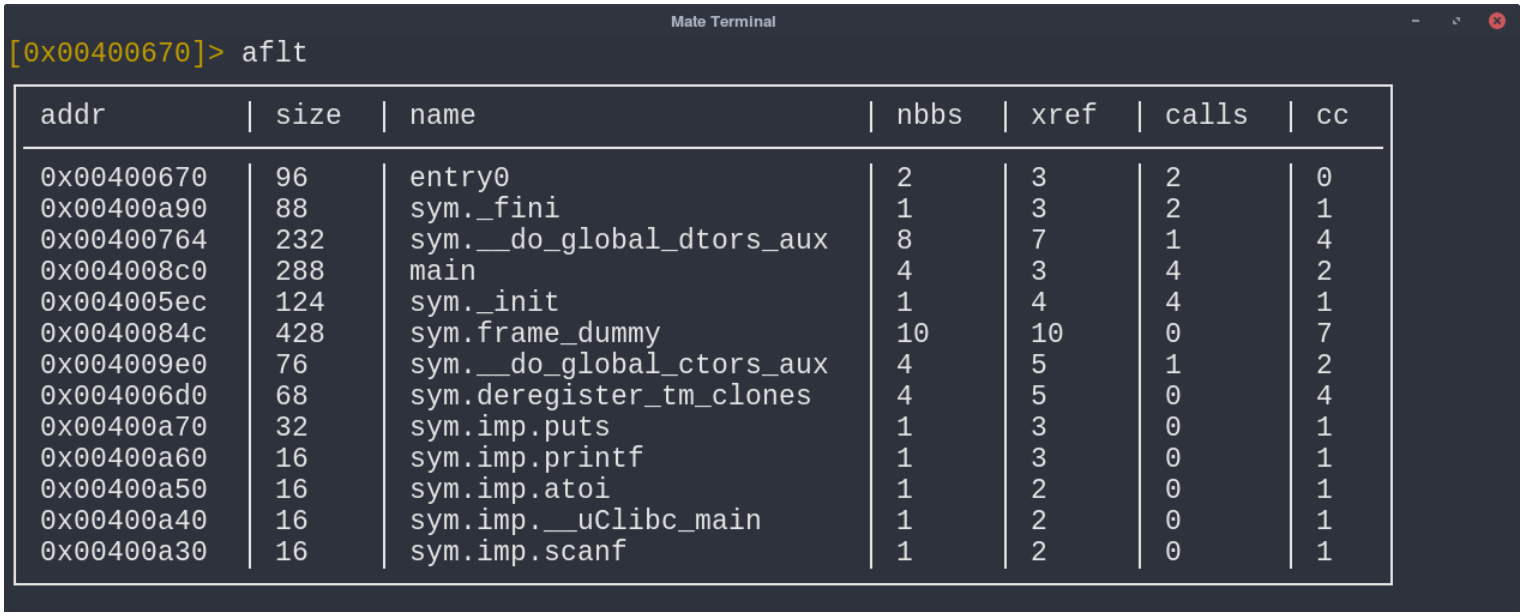
---

## Analysis

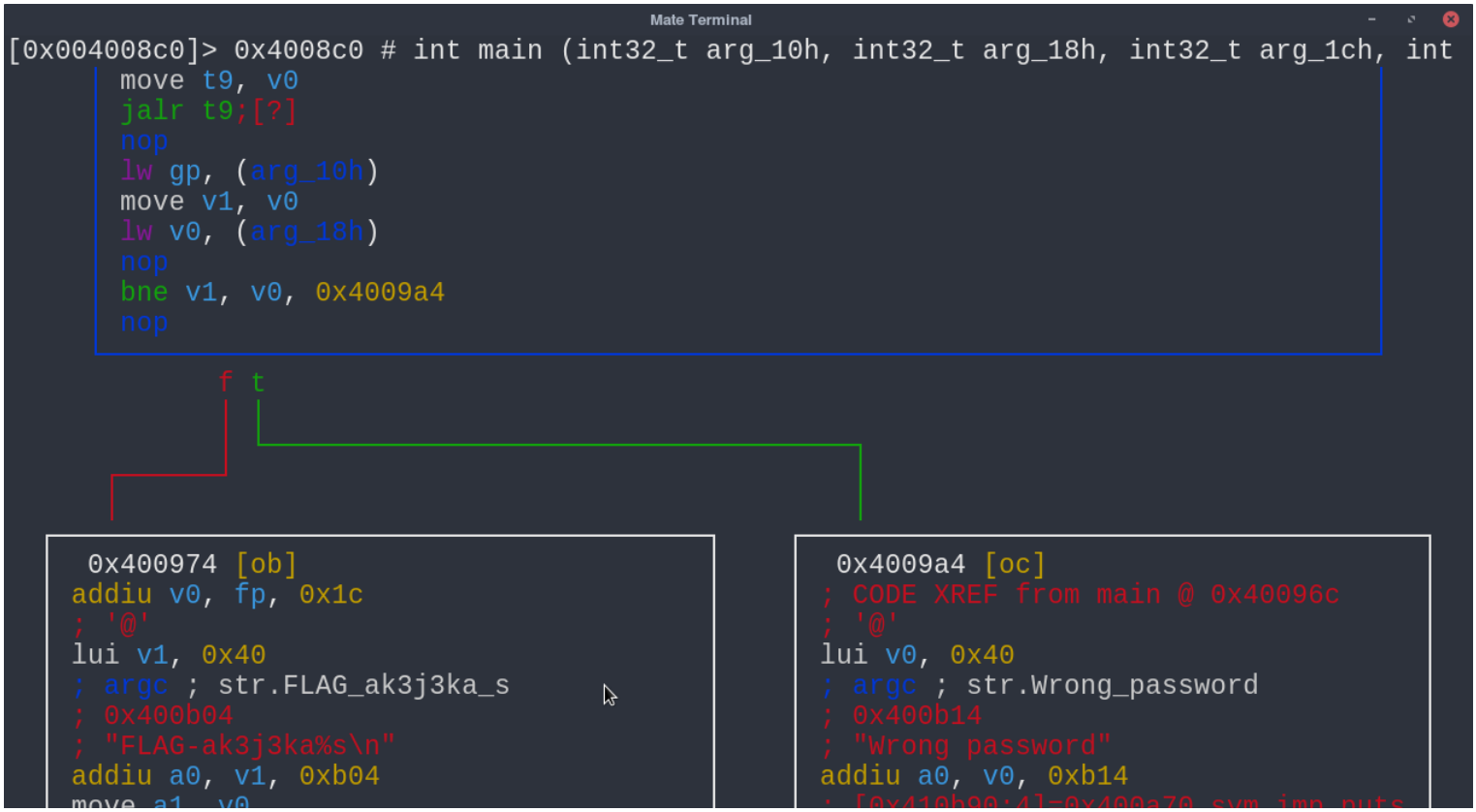
First we'll open and analyze the binary. We can see that its architecture is MIPS, which will prevent us from debugging it as normal (on my machine at least).



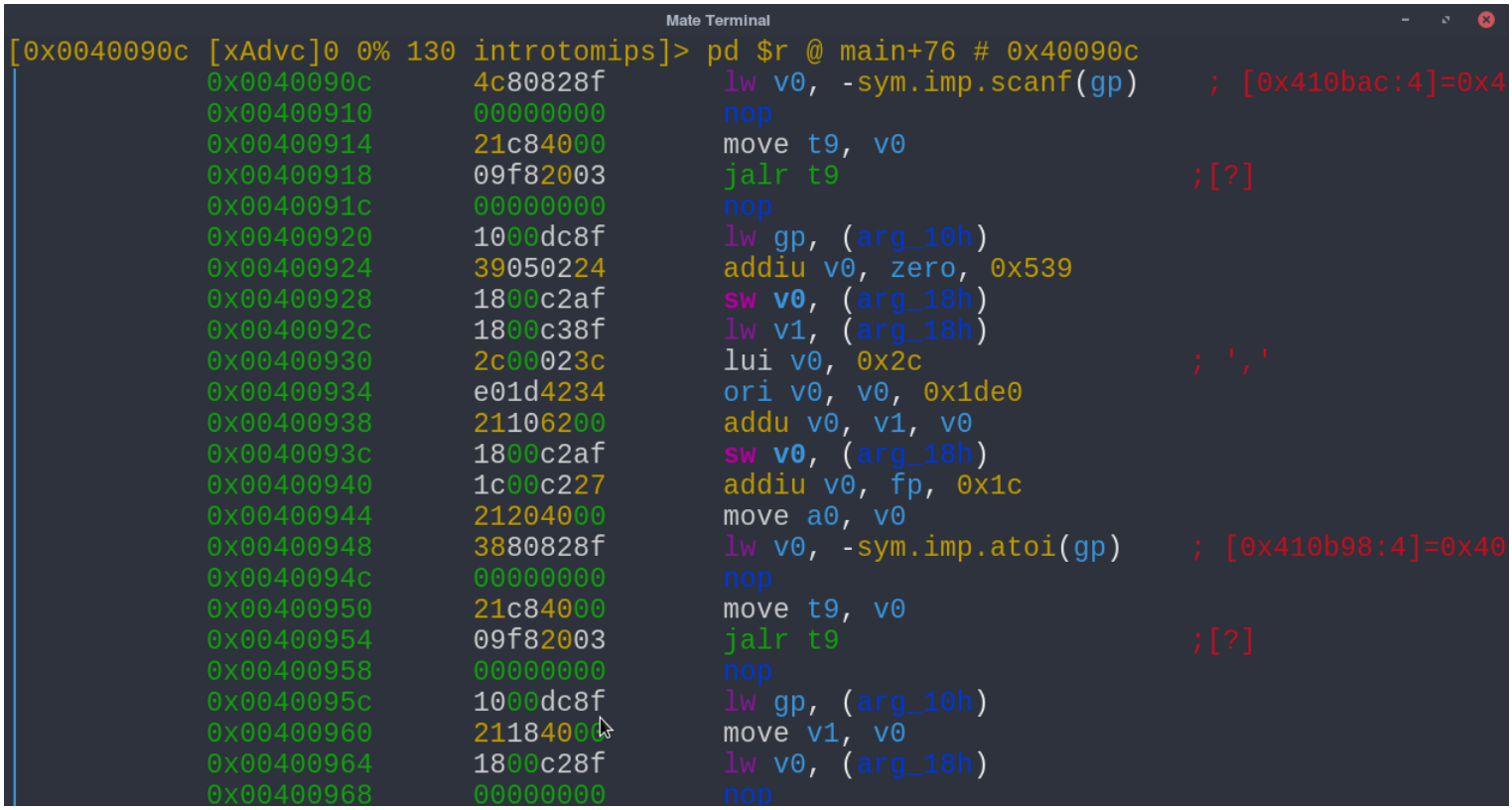

We can print a list of the functions, of which there are few, which will make solving this challenge much easier. Unsurprisingly, the key instructions can be found in the main function.



Towards the bottom of the main function there is a branch with what looks like a pass/fail condition, based on the user input earlier in the function. The condition tests for the equality of v1 and v0, so we'll look at how these are calculated.

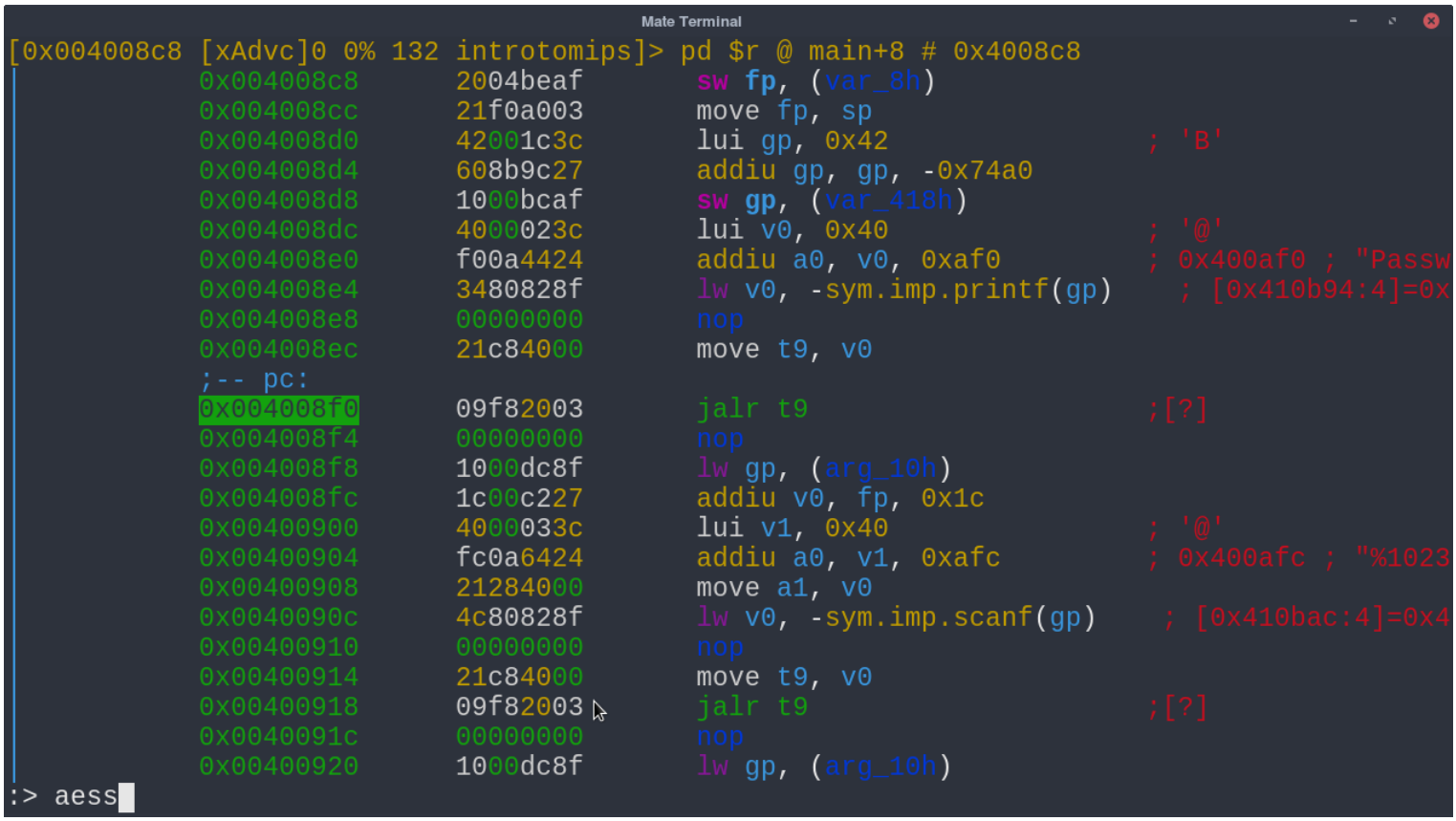



It looks like v0 is calculated with a simple bit of arithmetic, and is later compared to v1. We could do this by hand without too much trouble, but I'm lazy, so we'll try another way.

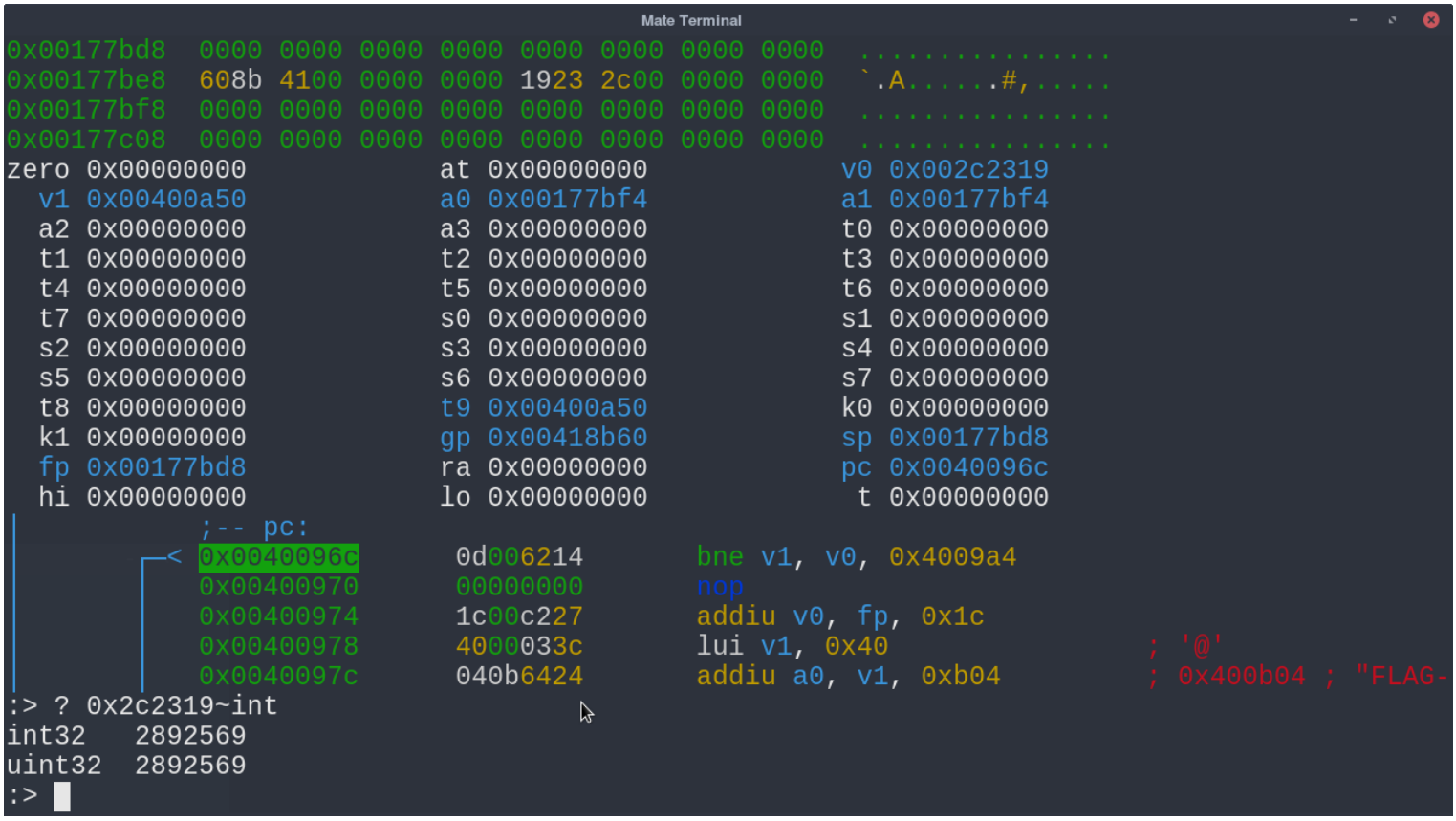


Using radare2's ESIL, we can emulate the instructions despite not being able to debug the binary like normal. First we'll seek to the main function and initialize the emulator using the three commands below. The first one starts the emulator, the second initializes the virtual memory, and the third initializes the program counter to the current seek address.


We can then step through the binary like normal using F7 or the aes/aeso commands. We'll want to skip over library calls using the aess command since the libraries aren't actually currently linked.




Once we get to the branch, examining the registers and stack shows that the v0 register is equal to 0x2c2319, which is equivalent to 2892569 in base 10.




If we combine this with the rest of the flag in the success branch, we get the solution:

**FLAG-ak3j3ka2892569**

EOF