**Task 4**

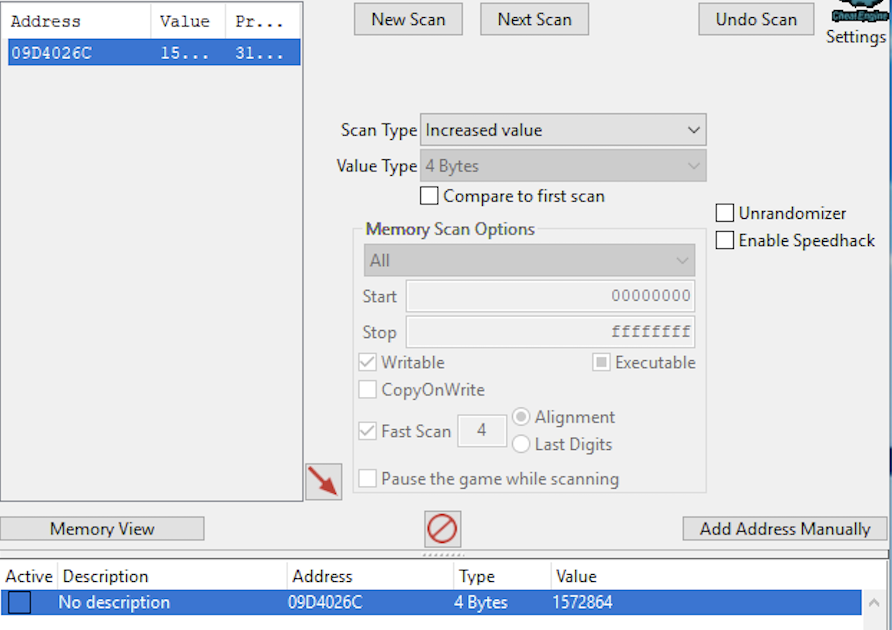
**Invulnerability**

**Team Members: Sayed Md Abu, Siyu Deng, Ige Tosin O**

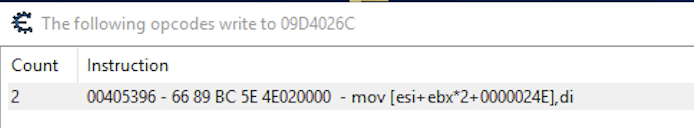
The process to accomplish this task is similar to the previous two. We need:

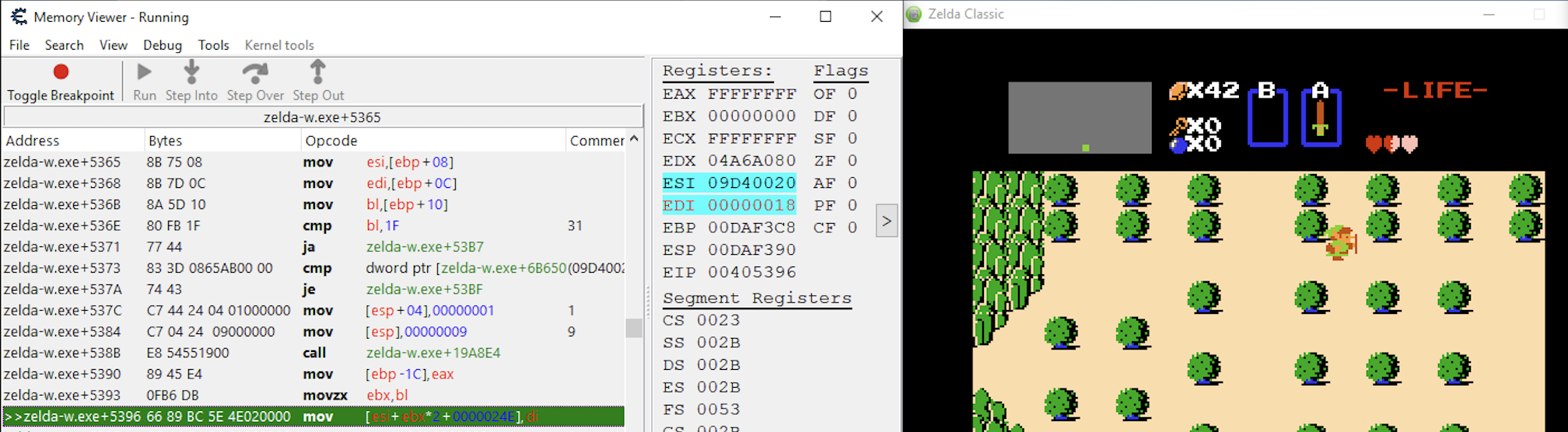
1. Find the memory address where tracks the health value;
2. Find what instruction writes to that value
3. Replace that value modification instruction (direct or indirect) with a constant preserving one.

So, the address of health value keeper is 09D4026C for this run. We find it by letting our character continuously losing health and scan decreased value after each losing. The process narrows down to one single address.



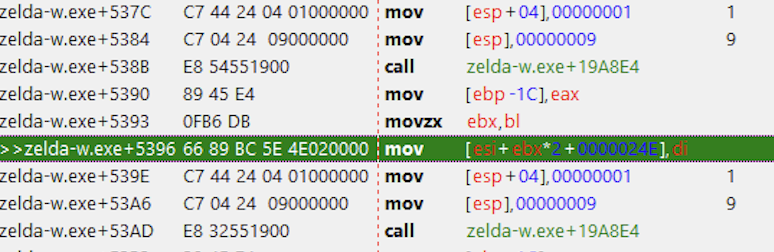
Then, we find which instruction modifies that value. And we noticed each time when our character get hurts, this function will be run once.



Now, let’s set a breakpoint at that instruction to see what value di carried when our character has full health, and also when it loses health. It turns out that when our character has full health, di carries 30h. If our character remains x full hearts and 1 half heart, the value carried by di would be 10\*xh+8h. Such as, when our character has 2.5 heart, di carries 28h.

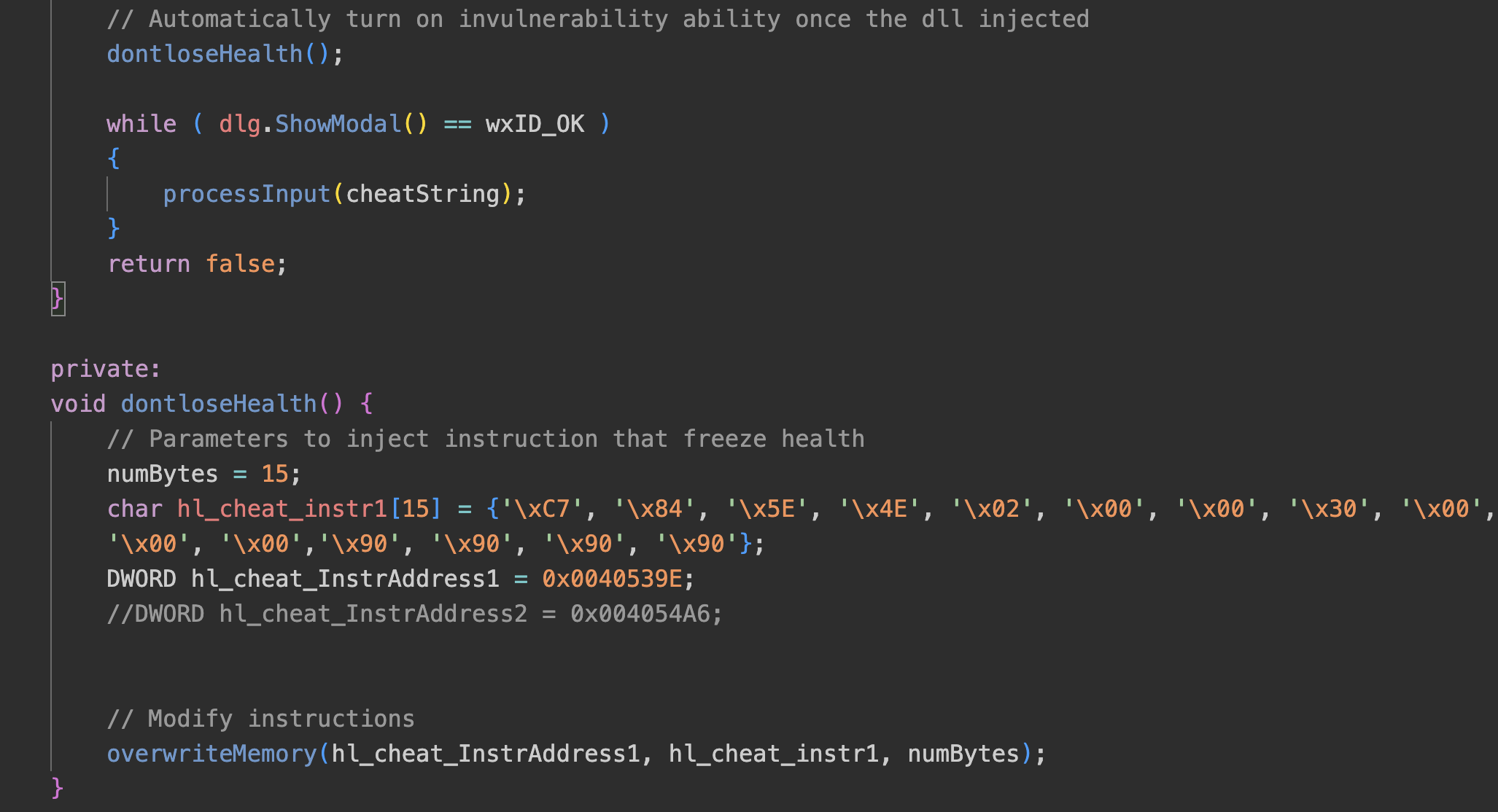
Similar as previous task, this instruction cannot be directly modified since it has no enough length. If we forcefully copy to that instruction, it will cause stack overflow which would automatically overwrite the following instructions. Then the whole program will be crashed.

By exploring near instructions, we find that instructions at 0040539E and 004053A6 are redundant. We can copy our instruction to these two addresses and use “nop” to fill up rest redundant spaces.



The instruction we would like to inject is “mov [esi+ebx\*2+0000024E],30” which is longer than the first instruction. For this approach, it is ok to overrun that address since the following instruction is also modifiable. Since after this injection there will still be 4 redundant bytes space, we fill them with ‘x90’.

Our code looks like:



And this functionality will be automatically enabled once the dll is injected.

Injectee.cpp is stored at **./Injectee/Injectee.cpp**.