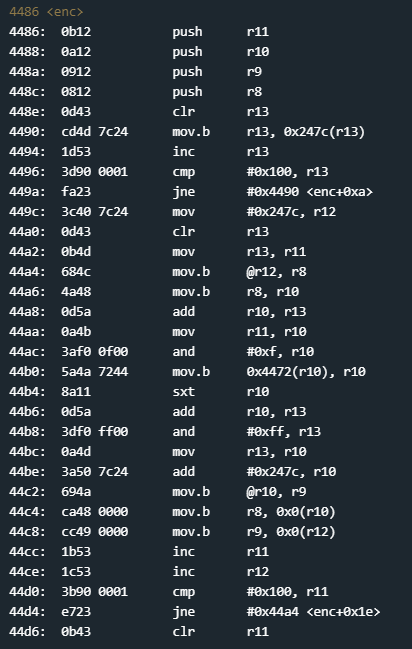
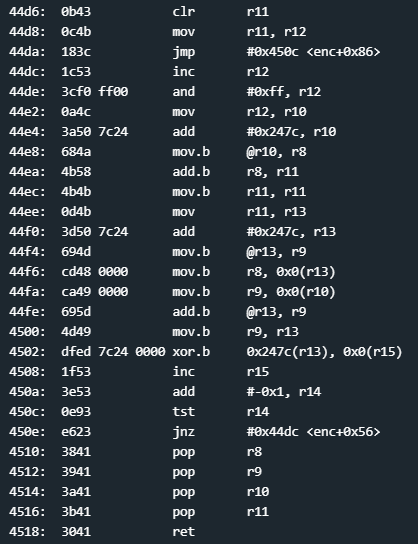
Unlike the last few challenges, we actually have a <main> function worth investigating:



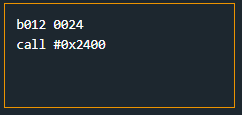
That seems pretty small, let’s check out that call to <enc>:

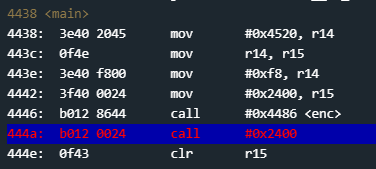


This here is a two parter:

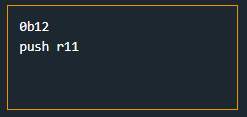


That is a lot of assembly. According to the manual for this lock, this function encrypts the password. Normally I would walk through this, but that call to #2400 in main looks interesting. Let’s set a breakpoint and go from there:





Aight, let’s take a step:



What? Where did that come from? If you look you cannot find a function #2400...so what is going on? That function call is to a direct memory location, that location being 0x2400. One of the many cool things about assembly is that you can execute instructions by their *opcode*.An opcode is the hexadecimal value of the instruction being executed. For example the opcode for the *call* instruction is b012. By calling a memory location we are telling the program to go and execute the instruction at that location. From there execution continues as normal. What this does is obfuscate the instructions so it is harder to reverse. Luckily, we can still see the instructions by stepping through the program. Here is what we get:

<#0x2400>

0b12 push r11

0412 push r4

0441 mov sp, r4

2452 add #0x4, r4

3150 e0ff add #0xffe0, sp

3b40 2045 mov #0x4520, r11

073c jmp $+0x10

6f4b mov.b @r11, r15

4f93 tst.b r15

f623 jnz $-0x12

1b53 inc r11

8f11 sxt r15

0f12 push r15

0312 push #0x0

b012 6424 call #0x2464

<#0x2464>

1e41 0200 mov 0x2(sp), r14

0212 push sr

0f4e mov r14, r15

8f10 swpb r15

024f mov r15, sr

32d0 0080 bis #0x8000, sr

bo12 1000 call #0x10

3241 pop sr

3041 ret

2152 add #0x4, sp

6f4b mov.b @r11, r15

4f93 tst.b r15

f623 jnz $-0x12

3012 0a00 push #0xa

0312 push #0x0

b012 6424 call #0x2464

<#0x2464>

1e41 0200 mov 0x2(sp), r14

0212 push sr

0f4e mov r14, r15

8f10 swpb r15

024f mov r15, sr

32d0 0080 bis #0x8000, sr

bo12 1000 call #0x10

3241 pop sr

3041 ret

2152 add #0x4, sp

3012 1f00 push #0x1f

3f40 d0ff mov #0xffde, r15

0f54 add r4, r15

0f12 push r15

2312 push #0x2

B012 6424 call #0x2464

<#0x2464>

1e41 0200 mov 0x2(sp), r14

0212 push sr

0f4e mov r14, r15

8f10 swpb r15

024f mov r15, sr

32d0 0080 bis #0x8000, sr

bo12 1000 call #0x10

3241 pop sr

3041 ret

3150 0600 add #0x6, sp

b490 59ed dcff cmp #0xed59, -0x24(r4) ← This is where the magic happens

0520 jnz $+0xc

3150 2000 add #0x20, sp

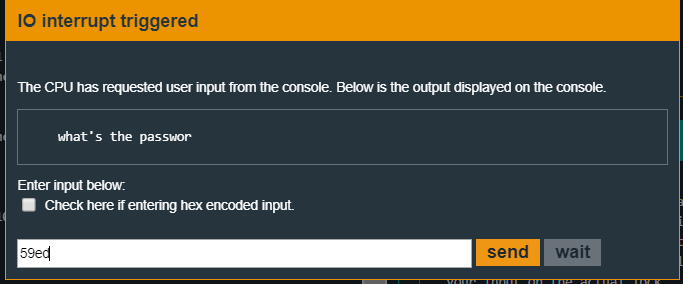
3441 pop r4

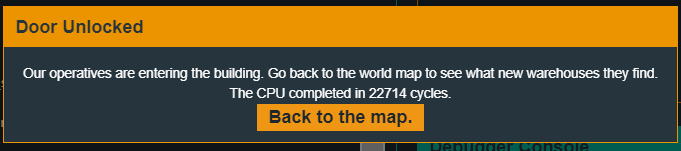
3b41 r11

3041 ret

[This is when it goes back to main]

That was A LOT of code, but we got through it. Like many of the other challenges, finding the cmp is our golden ticket. Let’s pass in the value 59ed, so that it is stored as ed59, and that should unlock the door:





Got it!