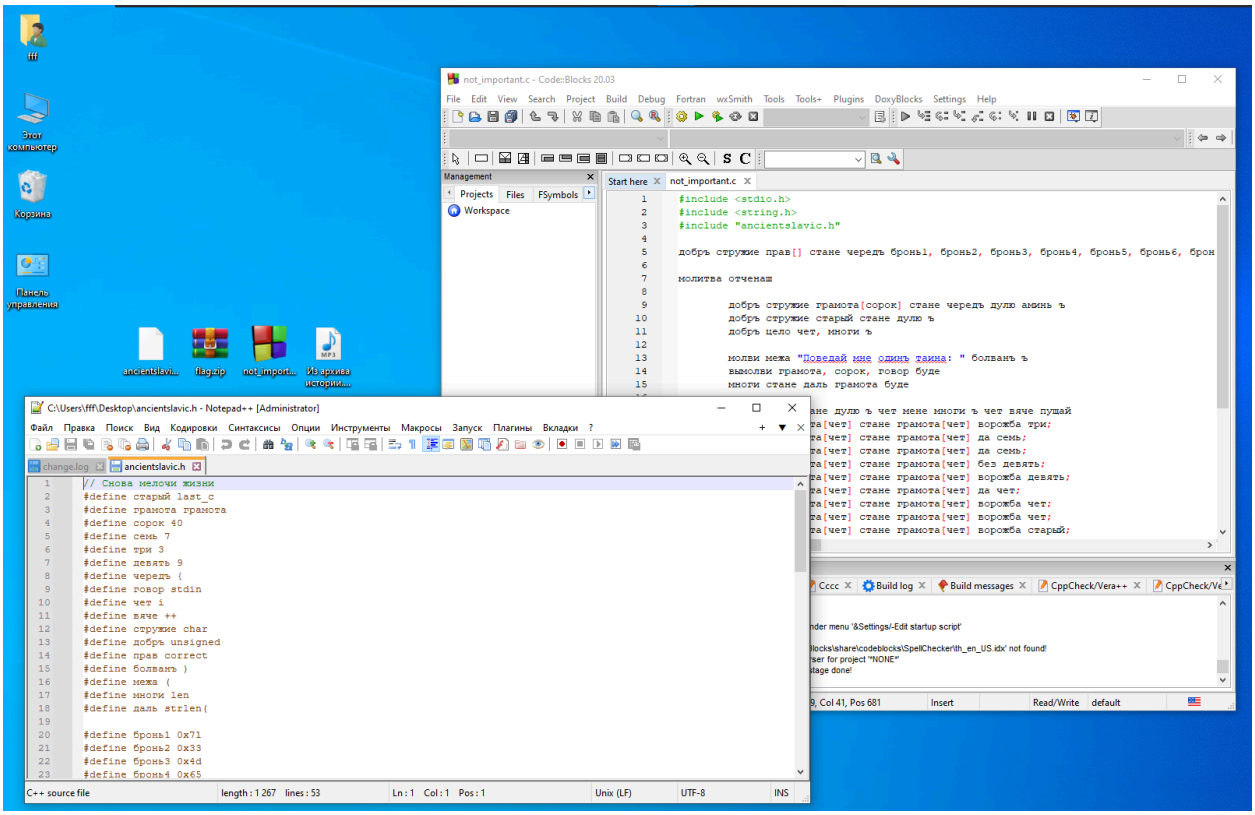
Название:	Таск древних русов
Категория:	Реверс-инжиниринг
Уровень:	Средняя
Очки:	550
Описание:	В архиве всё это время скрывалась правда о великой битве. А самое главное - сохранились файлы того времени! Только вот часть информации была утеряна...
Теги:	Побайтовое шифрование, патчинг файла
Автор:	ROP

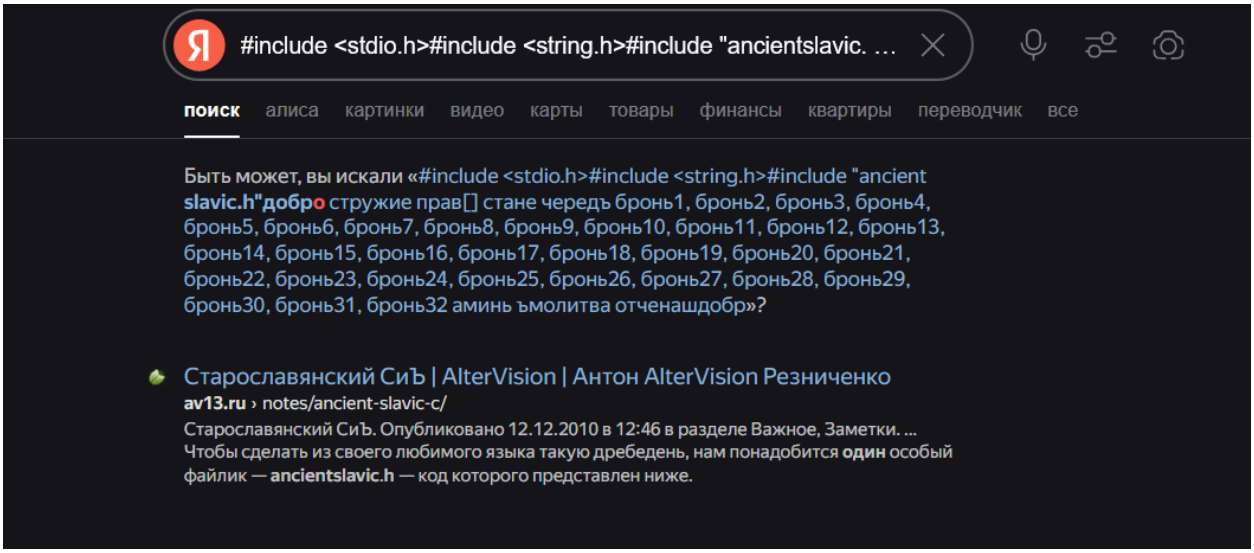
Прохождение:

Изучаем данные нам файлы.



Нам интересен Си-файл и Н-файл.

Если попробовать поискать этот язык по всему коду, то попадём сюда:



https://www.av13.ru/notes/ancient-slavic-c/

Тут есть дополнительные слова для H-файла:

```
// Файл: ancientslavic.h - переопределение языка

// Мелочи жизни
#define поболе >=
#define помене <=
#define боле >
#define мене <
#define аки ==
#define стане =
#define да +
#define без -
#define раз *
#define дели /
#define ъ ;
#define право true
#define бреше false
#define дулю 0

// if () { } else if () { } else { }
#define коли if(
#define пущай ){
#define ежели }else if(
#define либо }else{
```

```
#define аминь }
// коли а аки 1 пущай в стане 1 ежели а аки 2 пущай в стане 4 либо стане 0 аминь

// while () {} do{} while() for {}
#define покуда while(
#define твори do{
#define доколе }while(
#define буде );
#define откель for(
#define ступай goto
#define вон exit
#define
// покуда а мене 10 пущай а стане а да 1 аминь
// твори а стане а без 1 доколе а поболе 0 буде

// int float char etc
#define цело int
#define дробно float
#define передробно double
#define азъ char
#define непотребо void

// I/O
#define молви printf
#define поведай scanf

// Процедура - это функция, которая нам ничего не даёт, отсюда:
#define молитва int
#define мольба void
#define отченаш main(){
#define воздати return
// молитва отченаш воздати 0 ъ аминь
```

Можно заменить каждое слово из таска на значение из H-файла.

```
добръ стружие старый стане дулю ъ
добръ цело чет, многи ъ

молви межа "Поведай мне одинъ таина: " болванъ ъ
вымолви грамота, сорок, говор буде
многи стане даль грамота буде

откель чет стане дулю ъ чет мене многи ъ чет вяче пущай
        грамота[чет] стане грамота[чет] ворожба три;
        грамота[чет] стане грамота[чет] да семь;
        грамота[чет] стане грамота[чет] да семь;
        грамота[чет] стане грамота[чет] без девять;
        грамота[чет] стане грамота[чет] ворожба девять;
        грамота[чет] стане грамота[чет] да чет;
        грамота[чет] стане грамота[чет] ворожба чет;
        грамота[чет] стане грамота[чет] ворожба чет;
        грамота[чет] стане грамота[чет] ворожба старый;

        старый стане грамота[чет] ъ
аминь

коли быти грамота, прав, многи болванъ аки дулю пущай
```

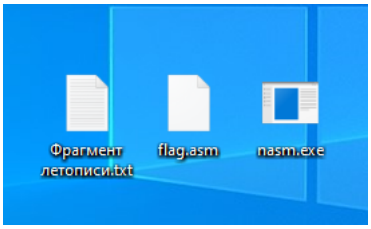
Например, вот тут будет шифрование введённого флага. Методом исключения выясним, что для слово `ворожба` подходит xor (так как его результат можно обратить, а +/- уже заняты).

Напишем код для дешифрования:

```
#include <stdio.h>
unsigned char correct[] = {
      0x71, 0x33, 0x4d, 0x65,
      0x11, 0x27, 0xa2, 0xee,
      0x9e, 0xf2, 0xc9, 0xa3,
      0xd7, 0x55, 0x32, 0x43,
      0x3b, 0x56, 0x6, 0x87,
      0xe0, 0x9b, 0xf9, 0x86,
      0xdf, 0x47, 0xc1, 0xb5,
      0xef, 0x7f, 0xff, 0xd9
};

int main() {
   int len = sizeof(correct);
   unsigned char last = 0;
   int i;

   for(i = 0; i < len; i++ ) {
```

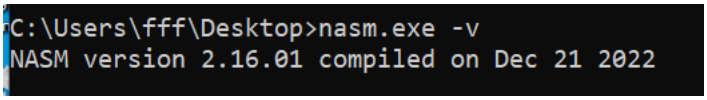
```
        unsigned char temp, j;

        for (j = 0x20; j < 128; j++) {
            temp = j;

            /* Алгоритм шифрования */
            temp = temp ^ 3;
            temp = temp + 7;
            temp = temp + 7;
            temp = temp - 9;
            temp = temp ^ 9;
            temp = temp + i;
            temp = temp ^ i;
            temp = temp ^ i;
            temp = temp ^ last;

            if (temp == correct[i]) {
                last = temp;
                printf("%c", j);
                break;
            }
        }
    }
    printf("\n");
    return 0;
}
```

Получили пароль от архива - `p@s$w0rD_f0R_tHe_S1aViC_@rcH1ve`.



Здесь `flag.asm` - это ложный вектор.

А вот `nasm.exe` пропатчен. Понять это можно, если скачать `nasm.exe` той же версии и сравнить его.



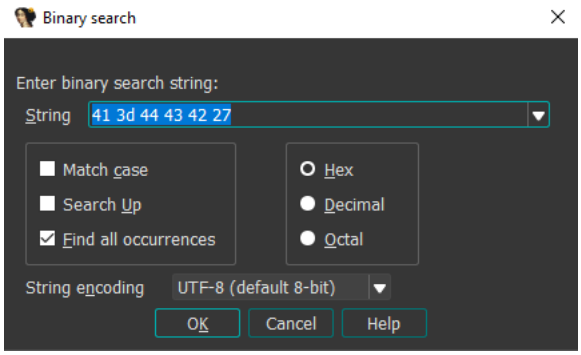
На сайте это:

https://nasm.us/pub/nasm/releasebuilds/2.16.01/win64/nasm-2.16.01-win64.zip

```
C:\Users\fff\Desktop>fc /b nasm.exe nasm_orig.exe
Сравнение файлов nasm.exe и NASM_ORIG.EXE
0004ECB0: 41 00
0004ECB1: 3D 00
0004ECB2: 44 00
0004ECB3: 43 00
0004ECB4: 42 00
0004ECB5: 27 00
0004ECB6: 49 00
0004ECB7: 3F 00
0004ECB8: 5E 00
0004ECB9: B0 00
0004ECBA: 9F 00
0004ECBB: 0F 00
0004ECBC: 63 00
0004ECBD: 3E 00
0004ECBE: 54 00
0004ECBF: 2D 00
0004ECC0: 54 00
0004ECC1: 40 00
0004ECC2: 51 00
0004ECC3: 39 00
0004ECC4: 2D 00
0004ECC5: 66 00
```

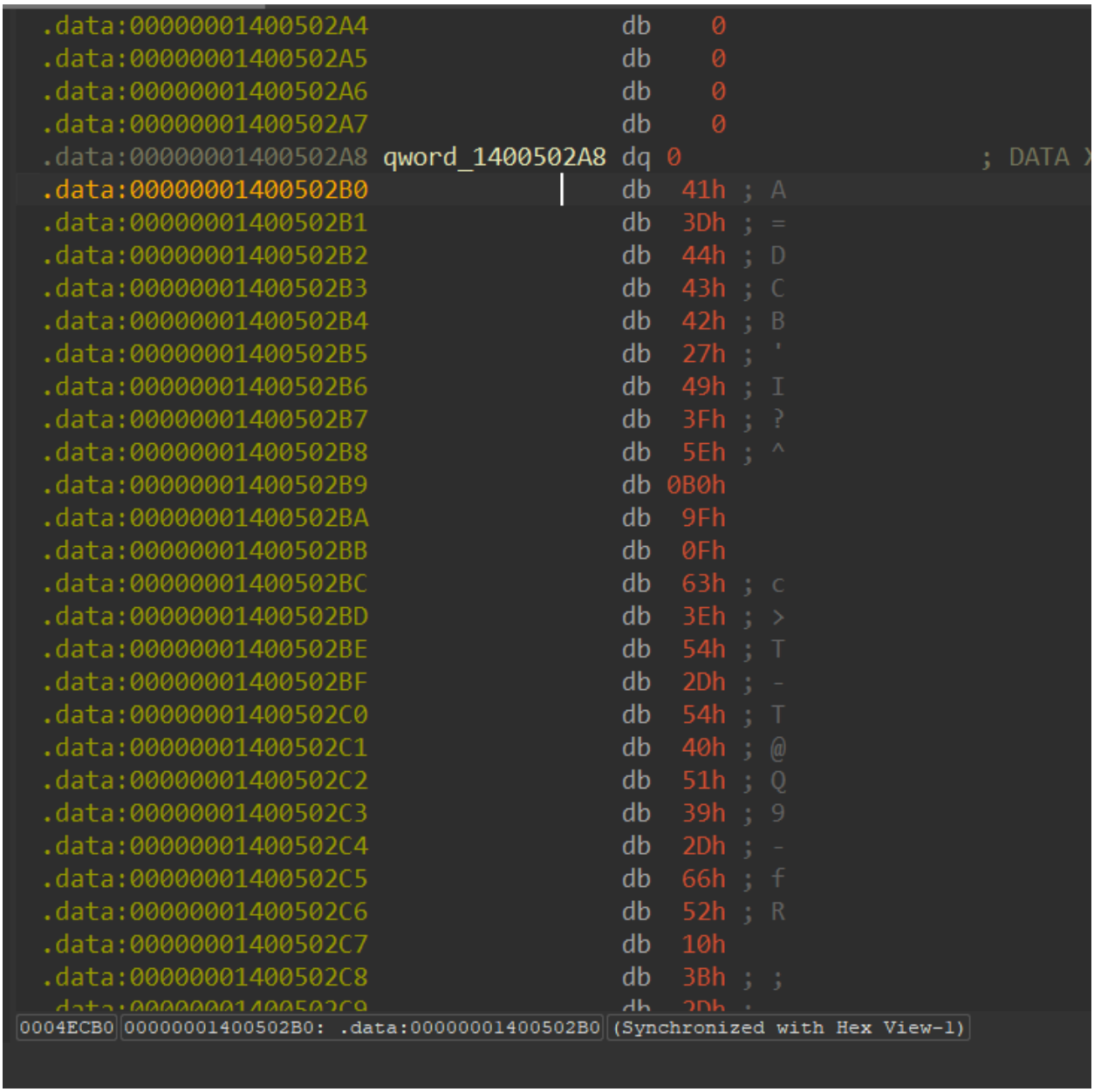
Найдём эти байты в файле через IDA.

Жмём ALT+B и ищем по:

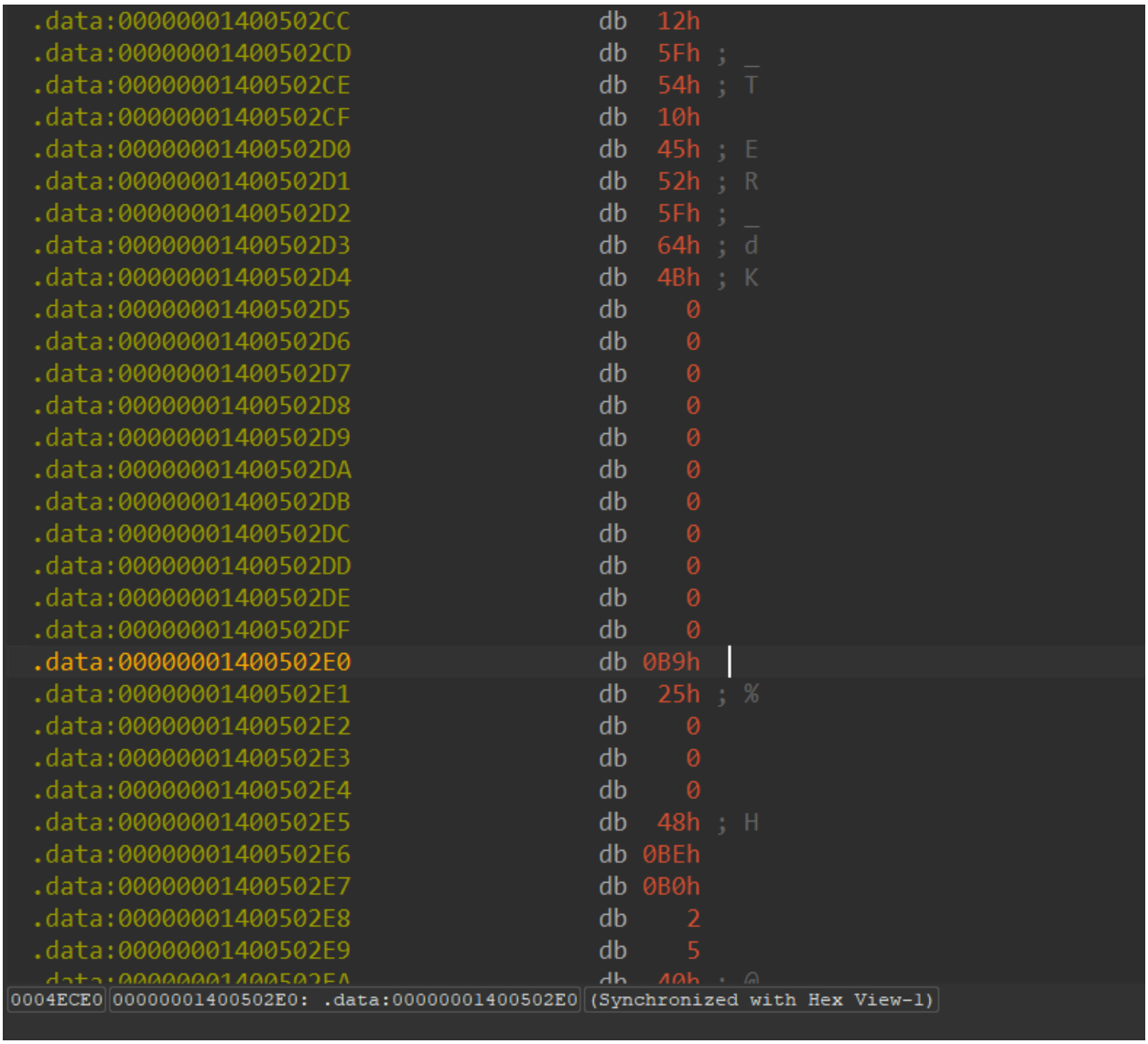


Или по другой последовательности байтов из данных выше.

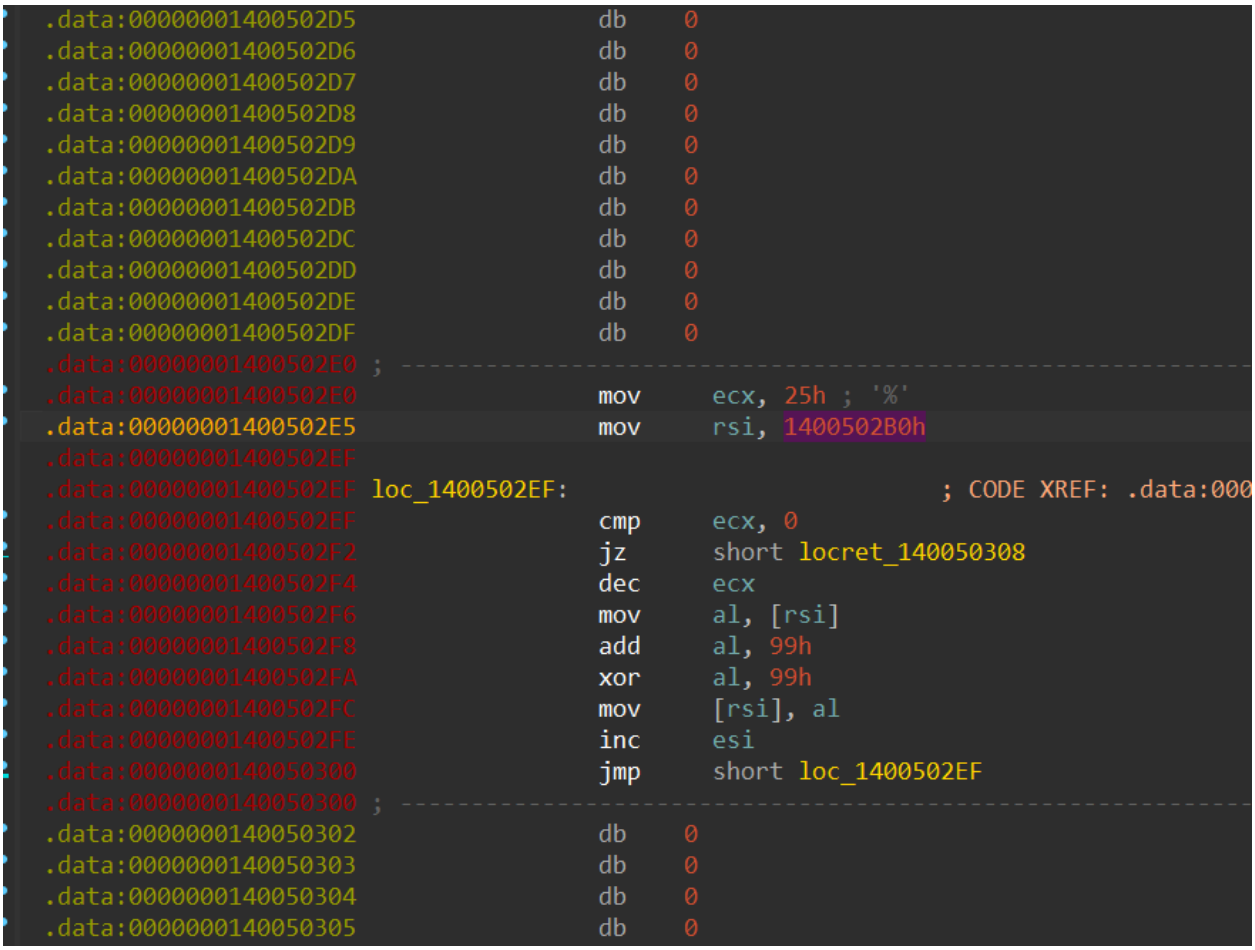
В итоге попадём сюда.



Можем попробовать представить эти байты как код, но ничего не выйдет. Нужно спуститься ниже после некого пробела из нулей.



Ставим курсор как на скрине и жмём “C”.



Этот код дешифрует байты выше.

Можем сделать это сами:

https://cyberchef.org/#recipe=From_Hex('Auto')ADD({'option':'Hex','string':'99'})XOR({'option':'He

В флаге остаёся непонятная часть.

CODEBY{AnÐ¡1eNt_

На самом деле это из-за кодировки. Попробовав разные кодировки поймём, что тут русский символ С.

Тогда начало флага будет:

```
CODEBY{AnC1eNt...
```