Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Отчет по домашнему заданию № 3

Выполнила: студентка 4 курса
Группы 171-341
Решетникова Дарья Алексеевна

Преподаватель:
Недогарок Антон Александрович

Куратор проекта: _____ / _____,
_____ /
подпись *ФИО, уч. звание и степень*

Студент: _____ / _____,
_____ /
подпись *ФИО, группа*

Москва 2020 г.

Вариант №13

Решение

1. Найти и привести в отчёте фрагмент кода, в котором происходит перенос введённого пользователем ключа в память crackme.
    a. Пометить начало и конец фрагмента комментариями в дизассемблере.

```
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cout
lea     rdx, aEnterKey  ; "Enter key:\t" - начало переноса введённого пользователем ключа в память crackme
call    sub_7FF60FDD1280
mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char,std::char_traits<char>> std::cin
lea     buffer_key, [rsp+258h+var_118]
lea     rdx, [rbx-1]
call    input_key       ; конец
xorps   xmm0, xmm0
mov     [rsp+258h+var_19], bl
lea     rax, a68086871  ; "68-08-68-71"
movsd   [rsp+258h+var_228], xmm0
lea     r9, a12941365   ; "12-94-13-65"
mov     [rsp+258h+var_238], rax
lea     buffer_key, [rsp+258h+var_118]
```

    b. Присвоить переменной, хранящей введённый ключ, название buffer_key.

Запускаю отладку и смотрю, в переменной r8 содержится введенный key, значит это и есть buffer_key.

```
--------------------------------------------------------------------------------
Moscow Polytechnic University
Reverse Engineering Exam

Serial: 207-780
Enter key:      111222333
```

```
xor     edx, edx        ; Val
lea     rcx, [rsp+258h+var_118] ; Dst
mov     r8d, 100h       ; Size
call    memset
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<c
lea     rdx, aEnterKey  ; "Enter key:\t"
call    sub_7FF60FDD1280
mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<ch
lea     r8, [rsp+258h+var_118]
lea     rdx, [rbx-1]
call    sub
xorps   xmm
mov     [rsp
lea     rax,
movsd   [rsp
lea     r9,
mov     [rsp
lea     r8,
lea     rdx,
lea     rcx,
call    sub
cmp     cs:byte_7FF60FDD6110, bl
jnz     short loc_7FF60FDD1C36
```

```
r8=Stack[000056E8]:0000008E518FF630
                db  31h ; 1
                db  31h ; 1
                db  31h ; 1
                db  32h ; 2
                db  32h ; 2
                db  32h ; 2
                db  33h ; 3
                db  33h ; 3
                db  33h ; 3
                db    0
```

Переименовываю:

```
lea     rax, a68086871  ; "68-08-68-71"
movsd   [rsp+258h+var_228], xmm0
lea     r9, a12941365   ; "12-94-13-65"
mov     [rsp+258h+var_238], rax
lea     buffer_key, [rsp+258h+var_118]
lea     rdx, [rsp+258h+Dst]
lea     rcx, aKeyCtf    ; "key = ***{CTF}***"
call    sub_7FF60FDD1010
cmp     cs:byte_7FF60FDD6110, bl
jnz     short loc_7FF60FDD1C36
```

```
std::cout
```

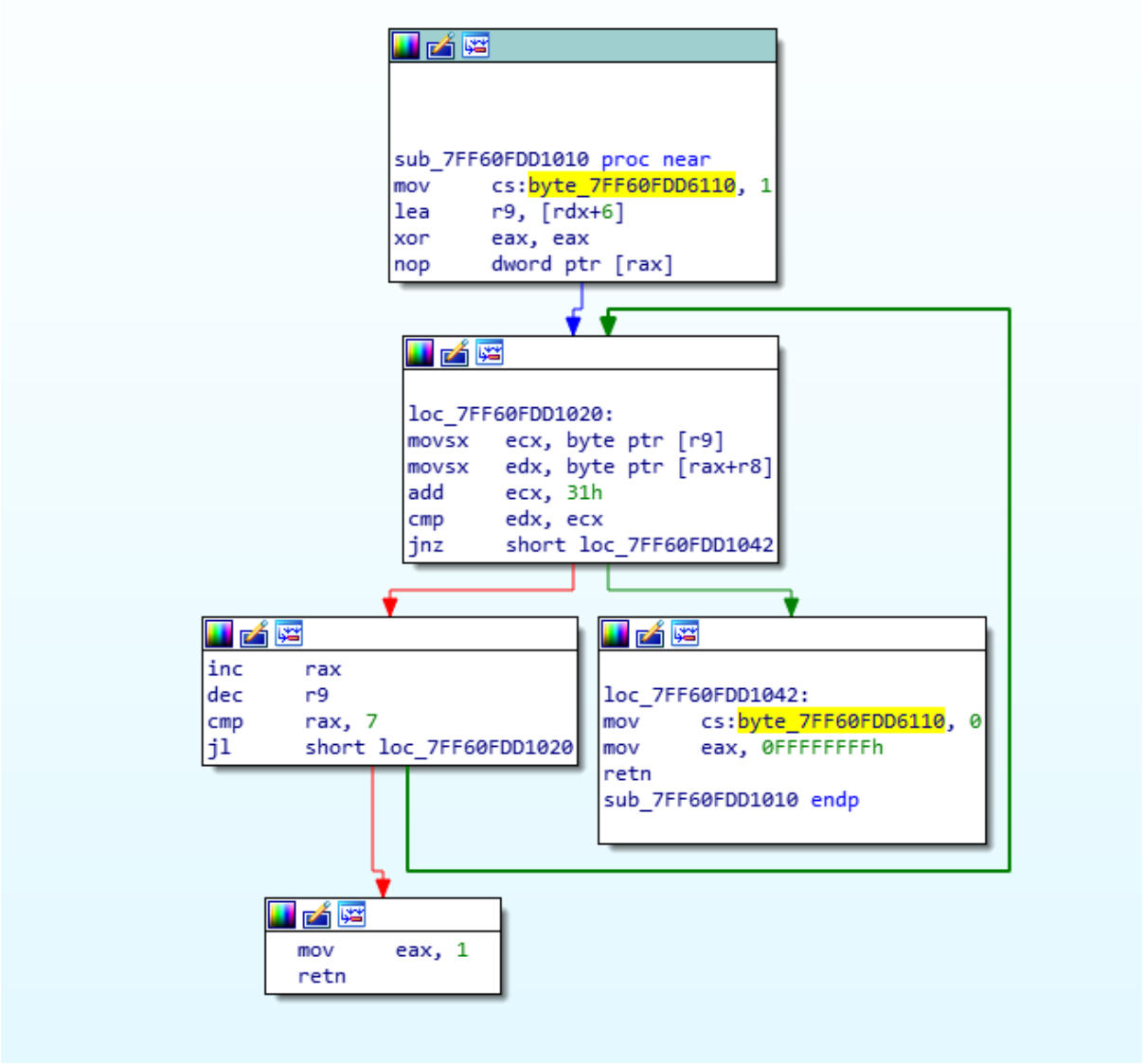
c. Привести снимок экрана дизассемблера в отчёте.

2. Найти и привести в отчёте фрагмент кода, в котором происходит проверка корректности ключа.
   a. Пометить начало и конец фрагмента комментариями в дизассемблере.

Проверка корректности ключа начинается здесь и заканчивается при JNZ (если сравнение не равно (JNZ) оно переносит к Activation failed =(. Exit, а если равно, то переносит к Successful key check!):

```
movsd   [rsp+258h+var_228], xmm0
lea     r9, a12941365   ; "12-94-13-65"
mov     [rsp+258h+var_238], rax
lea     buffer_key, [rsp+258h+var_118]
lea     rdx, [rsp+258h+Dst]
lea     rcx, aKeyCtf    ; "key = ***{CTF}***" - начало проверки корректности ключа
call    sub_7FF60FDD1010
cmp     cs:byte_7FF60FDD6110, bl
jnz     short loc_7FF60FDD1C36 ; конец
```

В этом фрагменте находится функция, открываем ее и убеждаемся, что именно этот фрагмент кода отвечает за проверку корректности ключа:

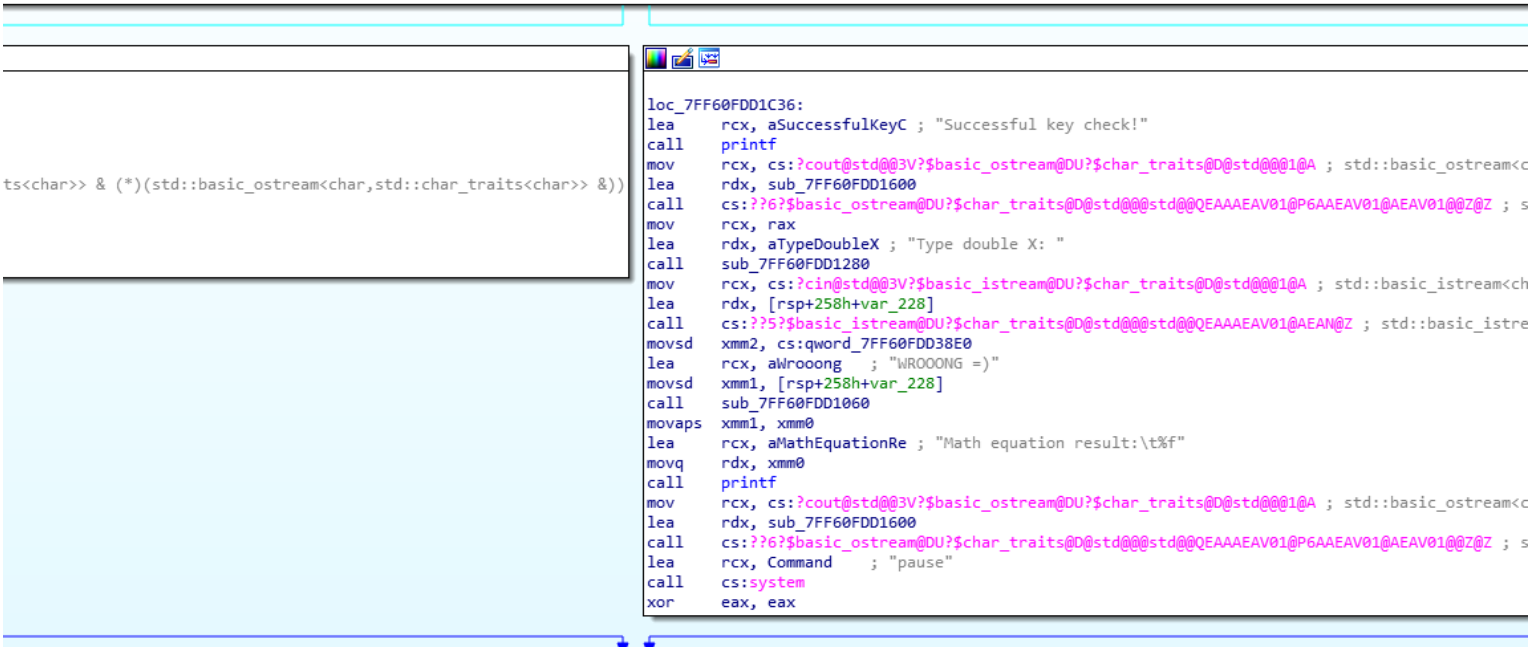


b. Привести снимок экрана дизассемблера в отчёте.

3. Найти и привести в отчёте фрагмент кода (ветку условного перехода), которая выполняется, если ключ верен.
    a. Пометить начало и конец фрагмента комментариями в дизассемблере.

По словам “Successful key check!” выясняем, что это именно этот фрагмент. И выполняется он, если сравнение равно. Далее запускается алгоритм для расчёта математической функции:

ки корректности ключа



b. Привести снимок экрана дизассемблера в отчёте.

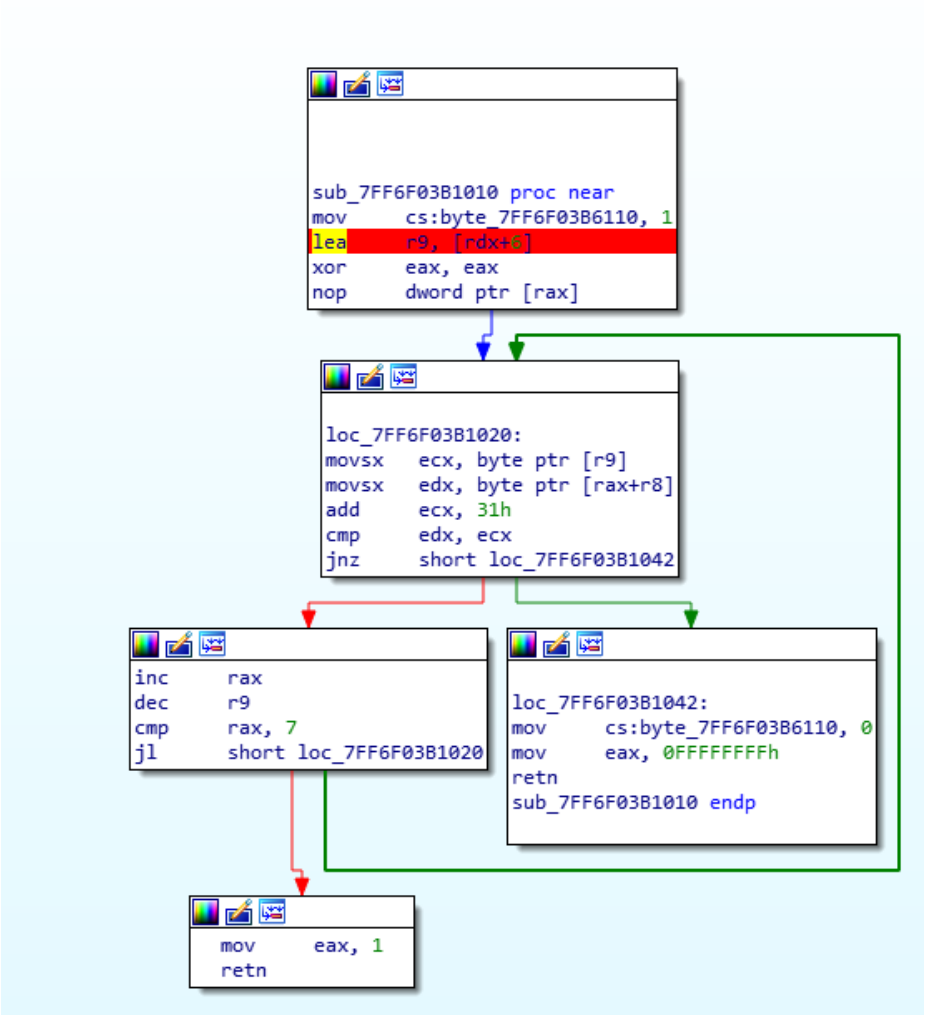
4. Найти и привести в отчёте фрагмент кода (ветку условного перехода), которая выполняется, если ключ не верен.
   a. Пометить начало и конец фрагмента комментариями в дизассемблере.

По словам “Activation failed =(. Exit” выясняем, что это именно этот фрагмент. И выполняется он, если сравнение не равно:

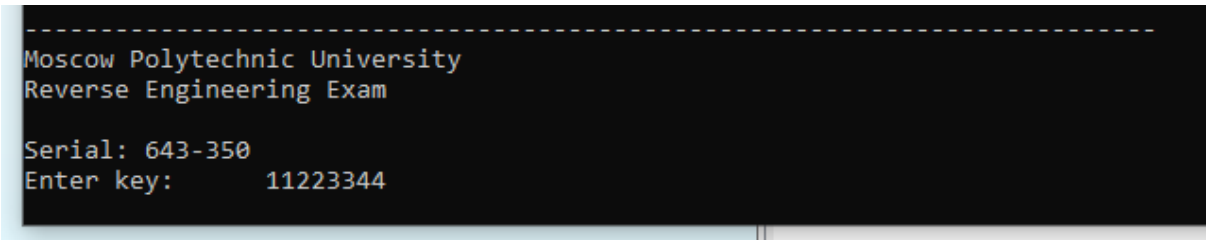
b. Привести снимок экрана дизассемблера в отчёте.

5. Описать в виде текста алгоритм, по которому происходит сверка ключа и серийного номера (обнаруженному в п.2). Развернуто прокомментировать фрагменты алгоритма сверки в дизассемблере и привести снимок экрана дизассемблера в отчёте.

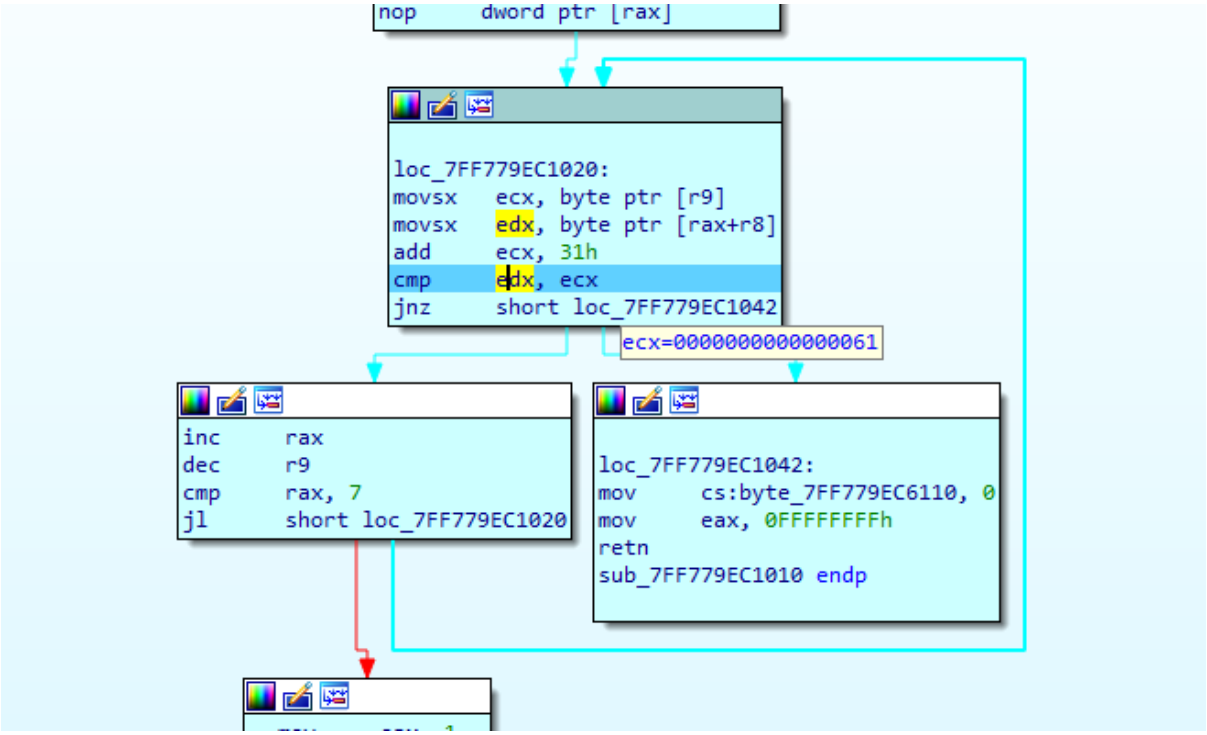
Для выяснения правильной комбинации воспользуемся отладкой. Ставим точку останова в начале функции и запускаем отладку:



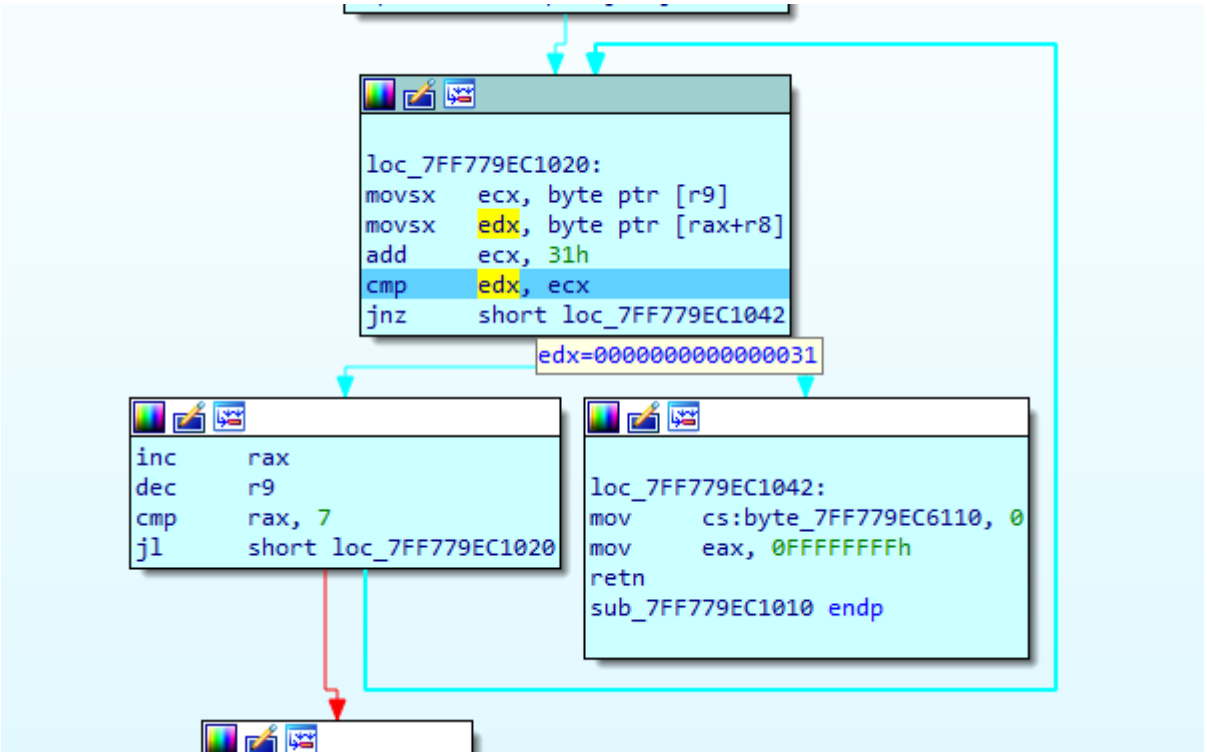
Введем рандомные значения:



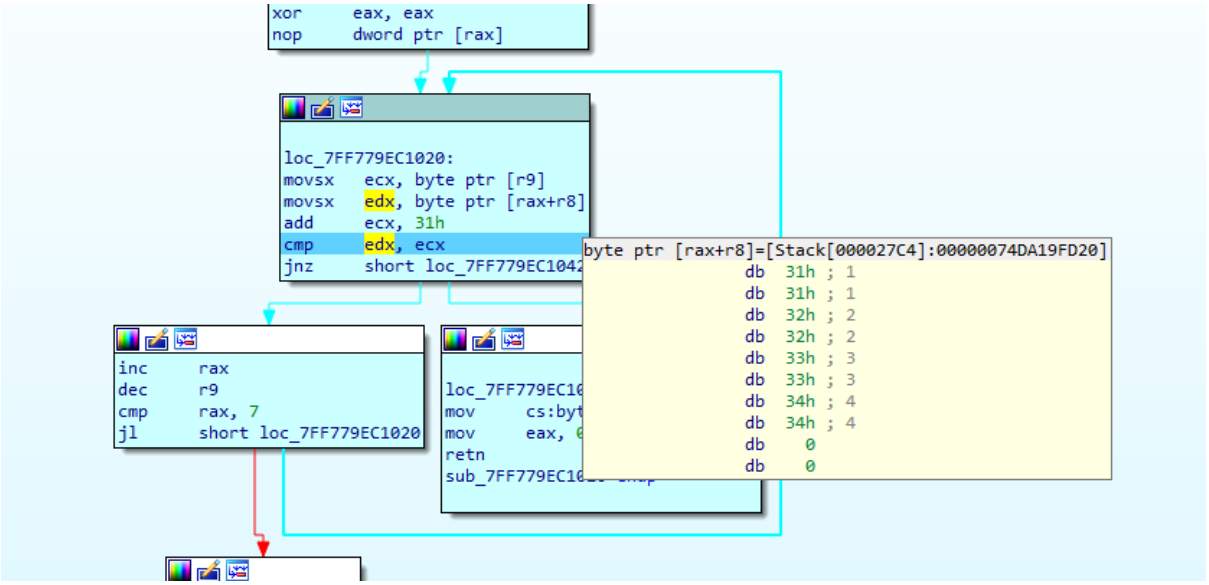
Видим, что регистр ecx = 61:



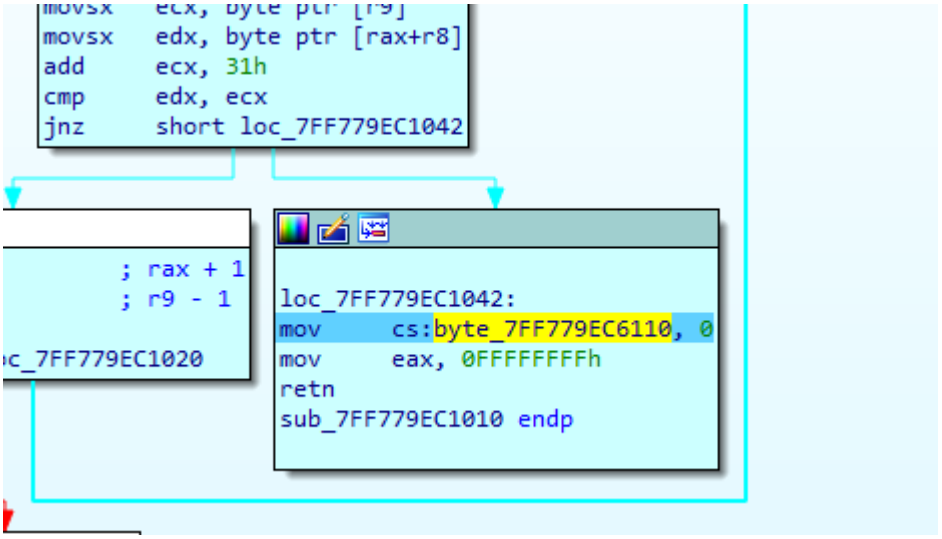
A edx (значения, которые мы вводили в key) = 31



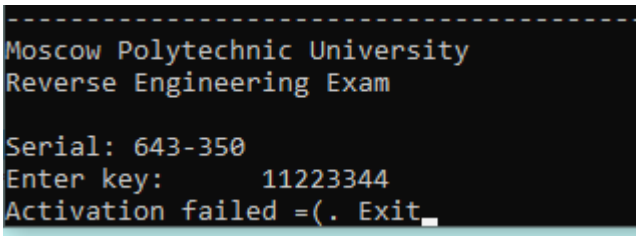
Edx сравнивается с нашим введенным key:



И нас выкидывает из функции, так как первое значение, очевидно, неверное.

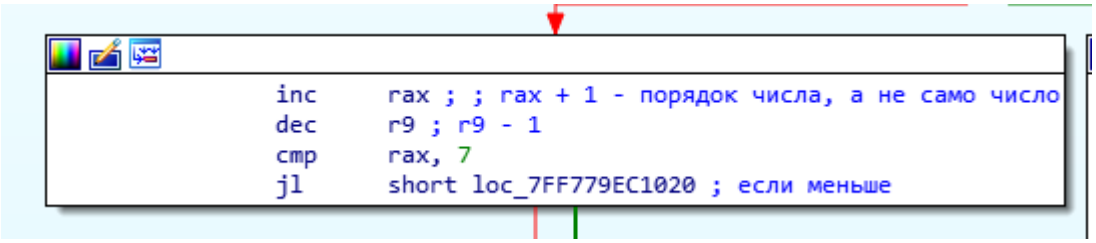


Выходит “Activation failed =(. Exit”:

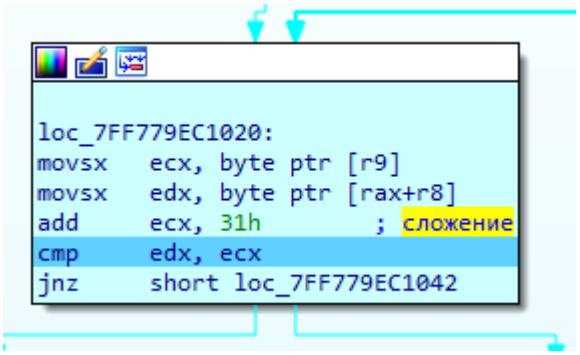


Делаем вывод, что если нас выкидывает в loc_7FF779EC1042 - значит пароль неверный. Рассмотрим другую ветку

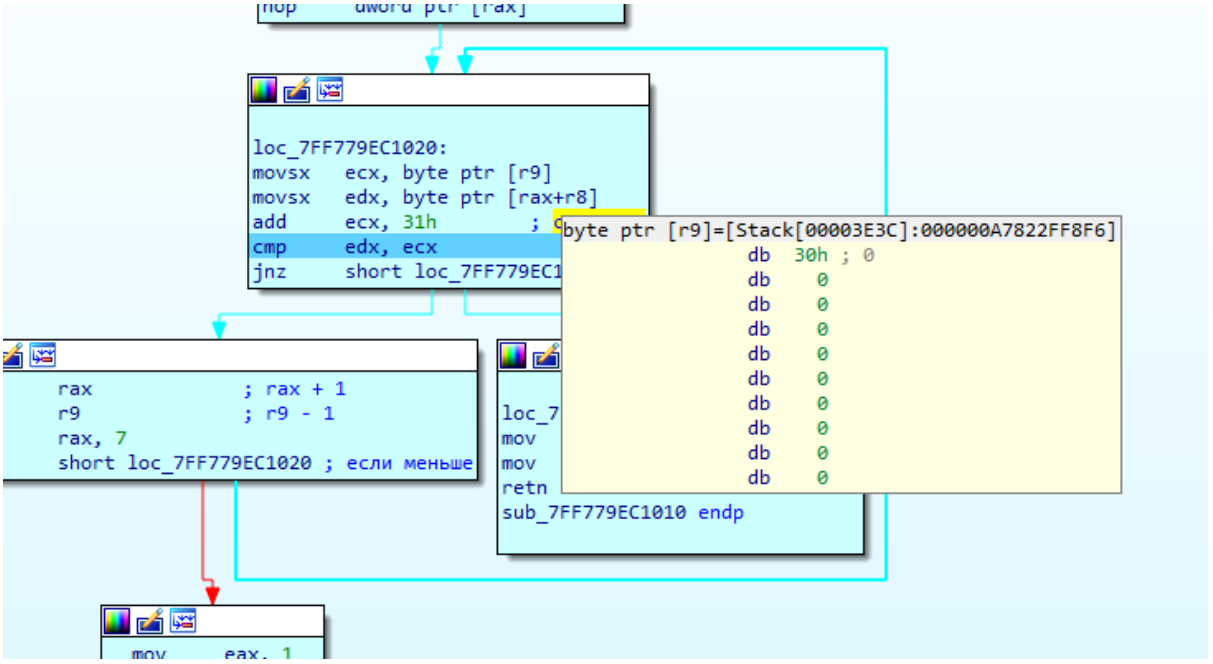
Тут мы видим, что rax сравнивается с 7, скорее всего в пароле будет 7 символов.



Рассмотрим алгоритм поподробнее, очевидно, что преобразование происходит в этой части кода:



Число 30h (в шестнадцатеричной системе)/ 0 (в десятичной) записывается в ecx. А потом к этому числу прибавляется 31h.



Получается 30h+31h=61h. Обратимся к таблице символов ASCII:

Hex	Char
20	
21	!
22	"
23	#
24	$
25	%
26	&
27	'
28	(
29	)
2A	*
2B	+
2C	,
2D	-
2E	.
2F	/
30	0
31	1
32	2
33	3
34	4
35	5
36	6
37	7
38	8
39	9
3A	:

Видим, что 30 в Hex = 0 в ASCII-символах (Char). Смотрим чему равен символ 61h:

Hex	Char
60	`
61	a

. Следовательно, в key используются ASCII-символы.

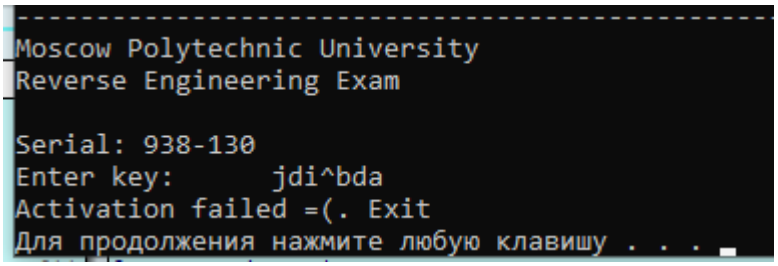
Попробуем запустить программу еще раз, в key на этот раз введем ASCII-символы, соответствующие каждой цифре в шестнадцатеричном пароле.

Dec	Hex	Char	Cmd
96	60	`	
97	61	a	
98	62	b	
99	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
107	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	

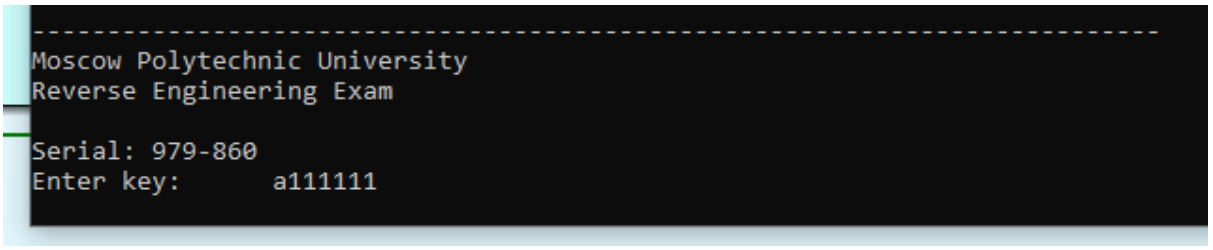
44	2C	,	
45	2D	-	
46	2E	.	

93	5D	]	
94	5E	^	
95	5F	_	

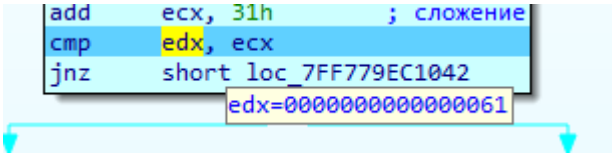
Однако попытка не увенчалась успехом =(.



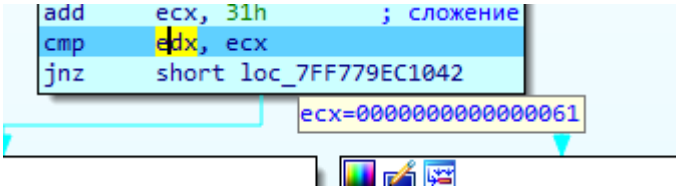
Смотрим дальше. Обратила внимание, что ecx сначала всегда равен 0. Нулю также равна всегда последняя цифра серийника. Можно попробовать написать первым символом a, а потом случайные цифры:



При отладке я обнаружила, что

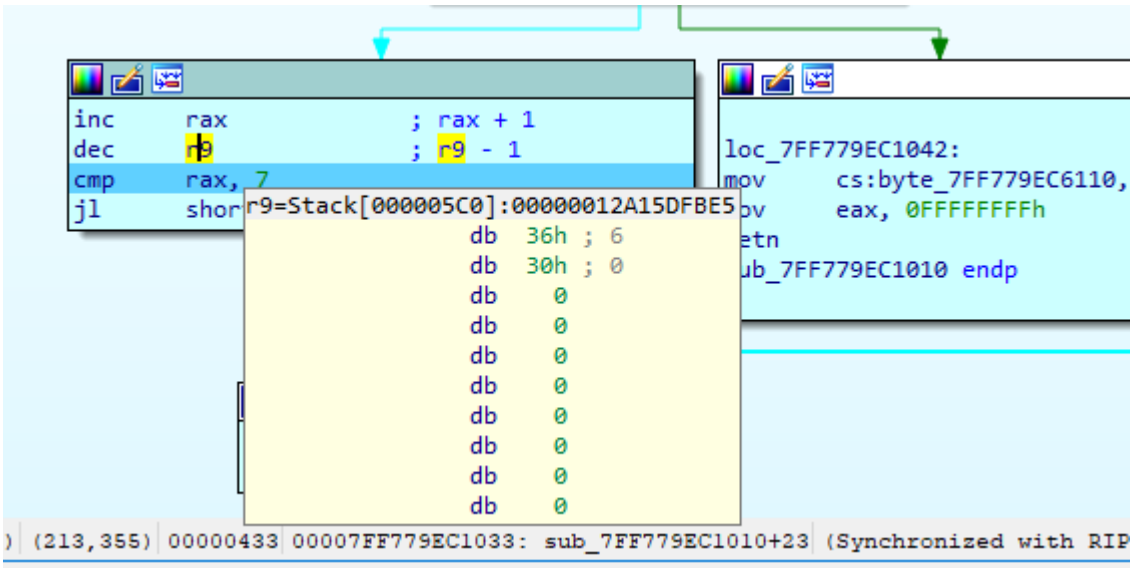


и



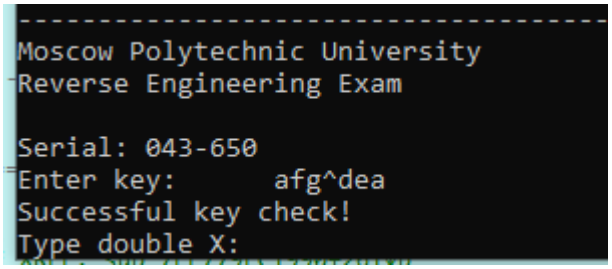
равны.

И действительно, у меня получилось войти в зацикленную ветку:



И при этом после в r9 (а это то, с чем сравнивается key) занесена 6. Точно также, как и в serial. Возможно, нужно перевернуть key задом наперед.

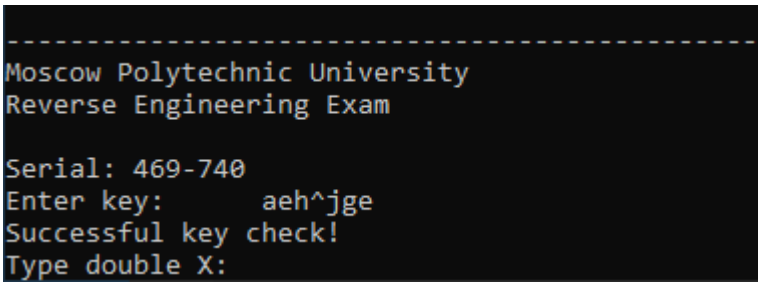
Пробую:

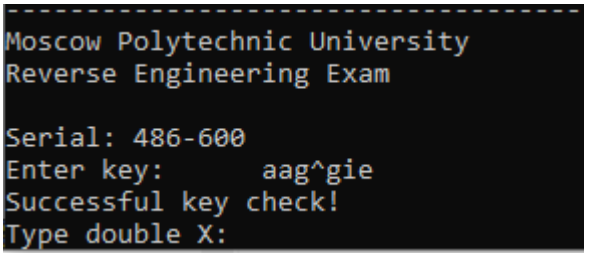


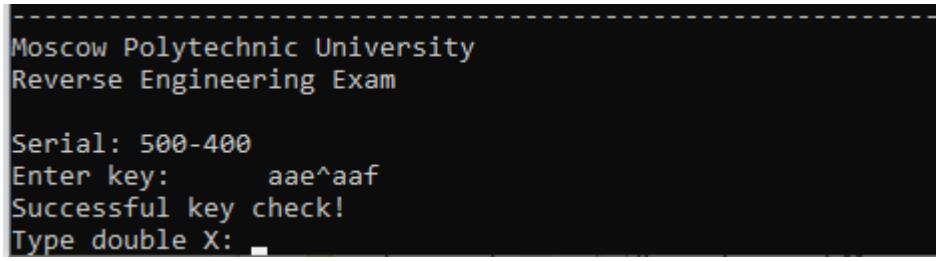
И действительно получилось.

Алгоритм: key - это ASCII-символы перевернутого шестнадцатеричного serial. Каждая цифра извлекается отдельно.

6. Привести как минимум 3 пары “ключ-серийный номер”, подобранные по алгоритму п.5, распознаваемые приложением как верные. Изменять код приложения (патчить) запрещено. Привести снимки экрана, подтверждающие корректную проверку пар “ключ-серийный номер”.







7. *Задание на дополнительные баллы*. По восстановленному алгоритму (п.5) написать на языке высокого уровня (рекомендуется C++ или Python) программу-keygen для подбора ключа по серийному номеру, предоставляемому самим crackme. Привести текст программы в отчёте. Рекомендуется реализовать keygen в как можно более простом коде (консольная программа, состоящая из одной функции и с использованием стандартных библиотек), чтобы проверить его функциональность можно было всего лишь скопировав текст keygen из отчёта в среду разработки.

```
key=input()[::-1]
for i in key:
    j=ord(i)+49
    i=+1
    print(chr(j))
```

```
123-269
j
g
c
^
d
c
b
```

key=input()[::-1]
for i in key:
    j=ord(i)+49
    i=+1
    print(chr(j))

Проверка:

```
----------------------------------
Moscow Polytechnic University
Reverse Engineering Exam

Serial: 582-570
Enter key:
```

```
582-570
a
h
f
^
c
i
f
```

```
Moscow Polytechnic University
Reverse Engineering Exam

Serial: 582-570
Enter key:      ahf^cif
Successful key check!
Type double X:
```

8. *Задание на дополнительные баллы*. В crackme также присутствует код для расчёта математической функции. Необходимо:
    a. Пометить начало и конец фрагмента комментариями в дизассемблере.

```
loc_7FF779EC1C36:
lea     rcx, aSuccessfulKeyC ; "Successful key check!"
call    printf
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cou
lea     rdx, sub_7FF779EC1600
call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::char_tra
mov     rcx, rax
lea     rdx, aTypeDoubleX ; "Type double X: " - начало математической функции
call    sub_7FF779EC1280
mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char,std::char_traits<char>> std::cin
lea     rdx, [rsp+258h+var_228]
call    cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAN@Z ; std::basic_istream<char,std::char_traits<char>>::ope
movsd   xmm2, cs:qword_7FF779EC38E0 ; начало расчёта математической функции (до этого был ввод значения)
lea     rcx, aWrooong ; "WROOONG =)"
movsd   xmm1, [rsp+258h+var_228]
call    sub_7FF779EC1060
movaps  xmm1, xmm0 ; Переслать выровненные упакованные короткие вещественные значения xmm1=xmm0 (а также конец самого расчета)
lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
movq    rdx, xmm0
call    printf ; конец
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cou
lea     rdx, sub_7FF779EC1600
call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::char_tra
lea     rcx, Command ; "pause"
call    cs:system
xor     eax, eax
```

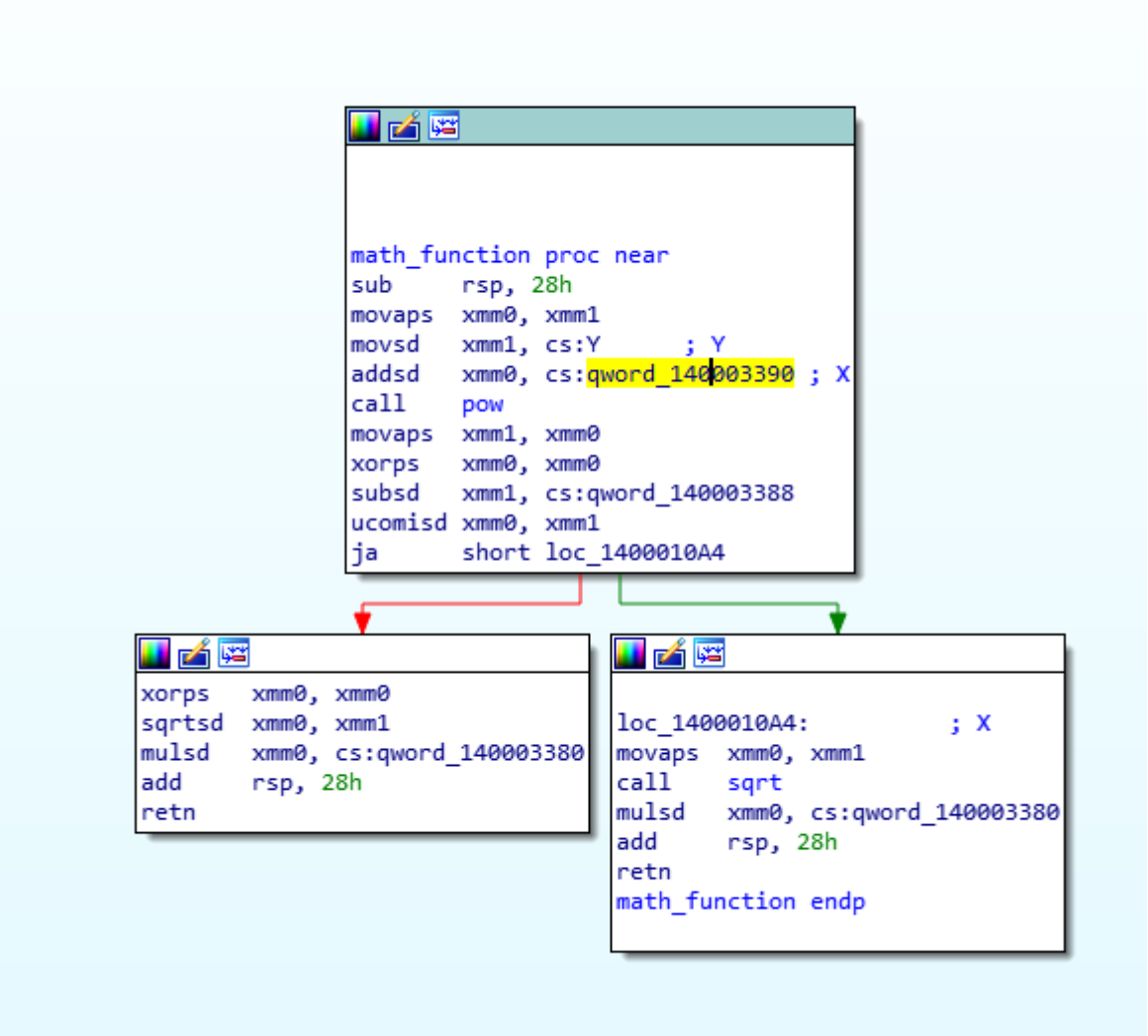
Очевидно, что все основные действия происходят в функции:

```
movsd   xmm2, cs:qword_7FF779EC38E0 ; Y
lea     rcx, aWrooong ; "WROOONG =)"
movsd   xmm1, [rsp+258h+var_228]
call    sub_7FF779EC1060
movaps  xmm1, xmm0 ; Переслать выровне
lea     rcx, aMathEquationRe ; "Math e
movq    rdx, xmm0
```

Переименую:

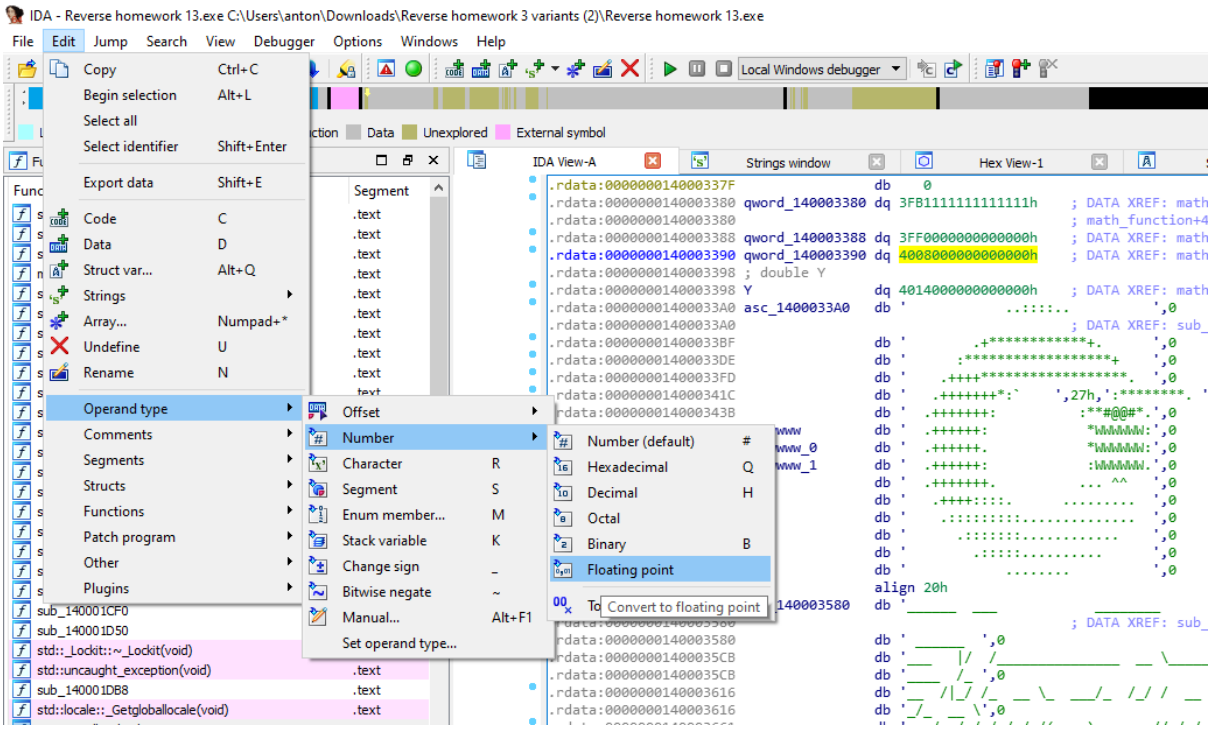
```
lea     rcx, aWrooong ; "WROOONG =)"
movsd   xmm1, [rsp+258h+var_228]
call    math_function
movaps  xmm1, xmm0 ; Переслать выров
lea     rcx, aMathEquationRe ; "Math
```

Функция выглядит так:



Преобразую числа:

```
addsd   xmm0, cs:qword_140003390 ; X
.rdata:0000000140003388 qword_140003388 dq 3FF0000000000000h    ; DATA XREF: math_function+22Fr
.rdata:0000000140003390 qword_140003390 dq 4008000000000000h    ; DATA XREF: math_function+F1r
.rdata:0000000140003398 ; double Y
```

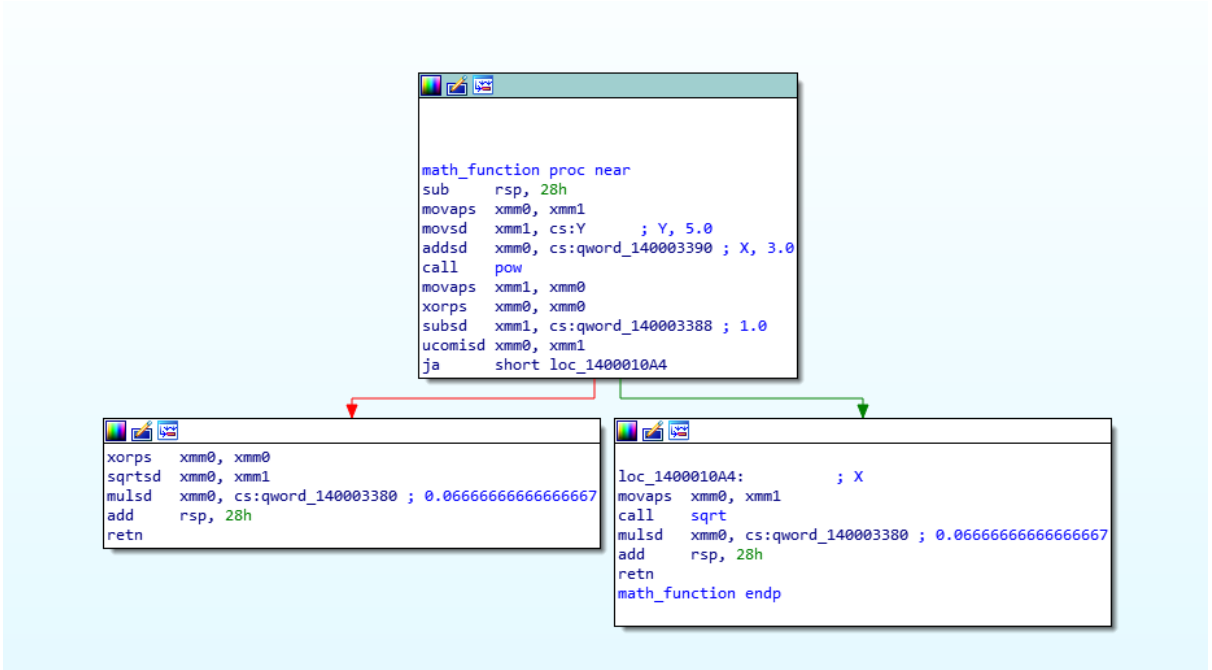


Также преобразую и остальные числа.

Y также преобразуется:

```
movaps  xmm0, xmm1
movsd   xmm1, cs:Y      ; Y
addsd   xmm0, cs:qword 14000339
```

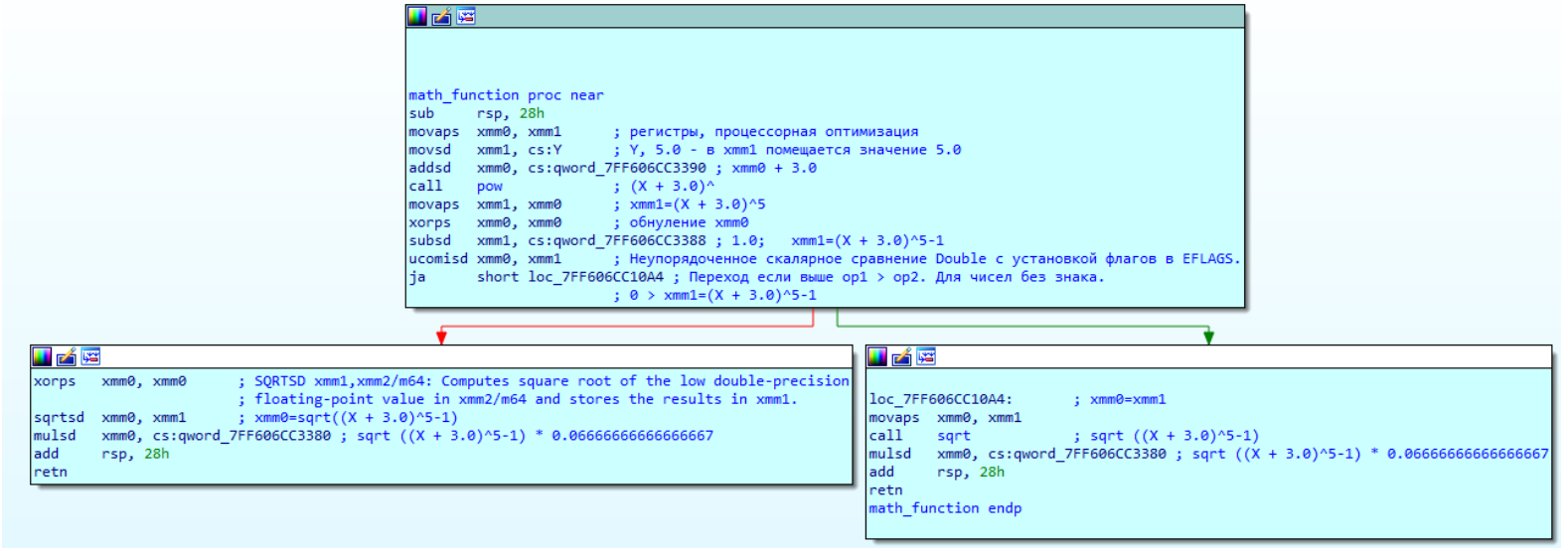
```
.rdata:000000014000337E                 db    0
.rdata:000000014000337F                 db    0
.rdata:0000000140003380 qword_140003380 dq 0.06666666666666667  ; DATA XREF: math_function+37↑r
.rdata:0000000140003380                                         ; math_function+4C↑r
.rdata:0000000140003388 qword_140003388 dq 1.0                  ; DATA XREF: math_function+22↑r
.rdata:0000000140003390 qword_140003390 dq 3.0                  ; DATA XREF: math_function+F↑r
.rdata:0000000140003398 ; double Y
.rdata:0000000140003398 Y               dq 5.0                  ; DATA XREF: math_function+7↑r
.rdata:00000001400033A0 asc_1400033A0   db '              ..::::..          ',0
.rdata:00000001400033A0                                         ; DATA XREF: sub_140001990+1F↑o
```



b. Записать функцию в отчёте в виде математического выражения.

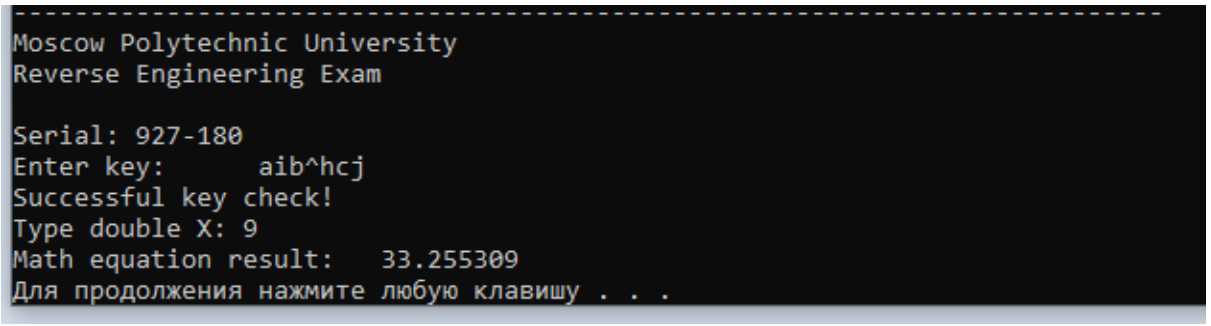
(X+3.0)^5-1
0 > (X+3.0)^5-1
Если нет, то:
sqrt((X+3.0)^5-1)*0.06666666666666667
Если да, то sqrt((X+3.0)^5-1)*0.06666666666666667

c. Пометить в дизассемблере в комментариях, каким именно образом строки диизассемблированного кода соотносятся с найденным математическим выражением. Должно быть наглядно и очевидно, как именно из кода на ассемблере получено выведенное математическое выражение.



Эта картинка есть в хорошем качестве называется “Math.png”.

d. Привести снимок экрана дизассемблера с кодом математической функции в отчёте.



(9+3)^5-1= 248 831
sqrt(248 831) * 0.06666666666666667 = 33,2553088

```
loc_7FF606CC1C36:
lea     rcx, aSuccessfulKeyC ; "Successful key check!"
call    sub_7FF606CC1CF0
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cout
lea     rdx, sub_7FF606CC1600
call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::char_traits<ch
mov     rcx, rax
lea     rdx, aTypeDoubleX ; "Type double X: "
call    sub_7FF606CC1280
mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char,std::char_traits<char>> std::cin
lea     rdx, [rsp+258h+var_228]
call    cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAN@Z ; std::basic_istream<char,std::char_traits<char>>::operator>
movsd   xmm2, cs:qword_7FF606CC38E0 ; в xmm2 помещается 15.1
lea     rcx, aWrooong   ; "WROOONG =)"
movsd   xmm1, [rsp+258h+var_228]
call    math_function
movaps  xmm1, xmm0
lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
movq    rdx, xmm0       ; Переслать 64 бита
call    sub_7FF606CC1CF0
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> std::cout
lea     rdx, sub_7FF606CC1600
call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::char_traits<ch
lea     rcx, Command    ; "pause"
call    cs:system
xor     eax, eax
```