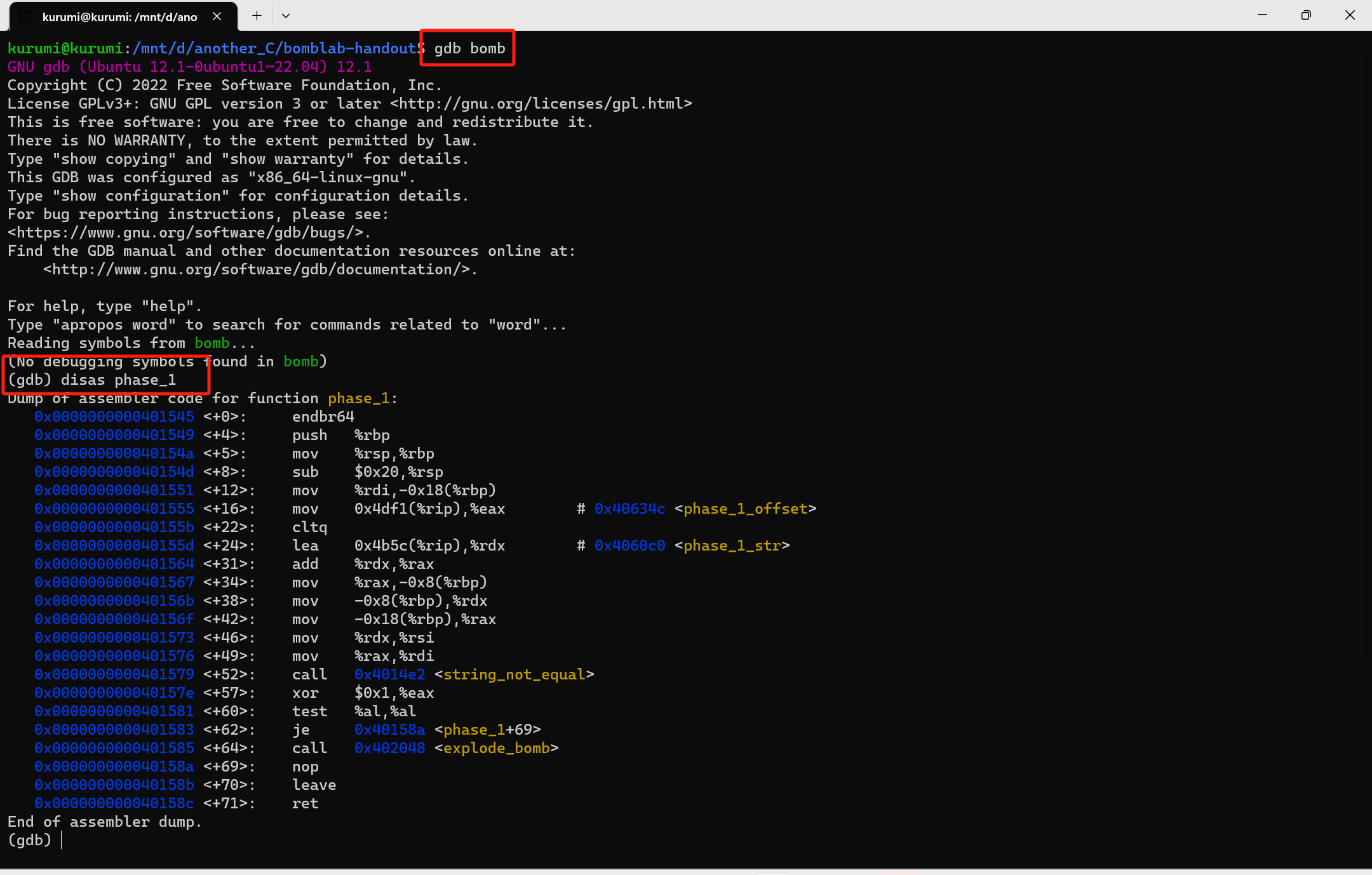
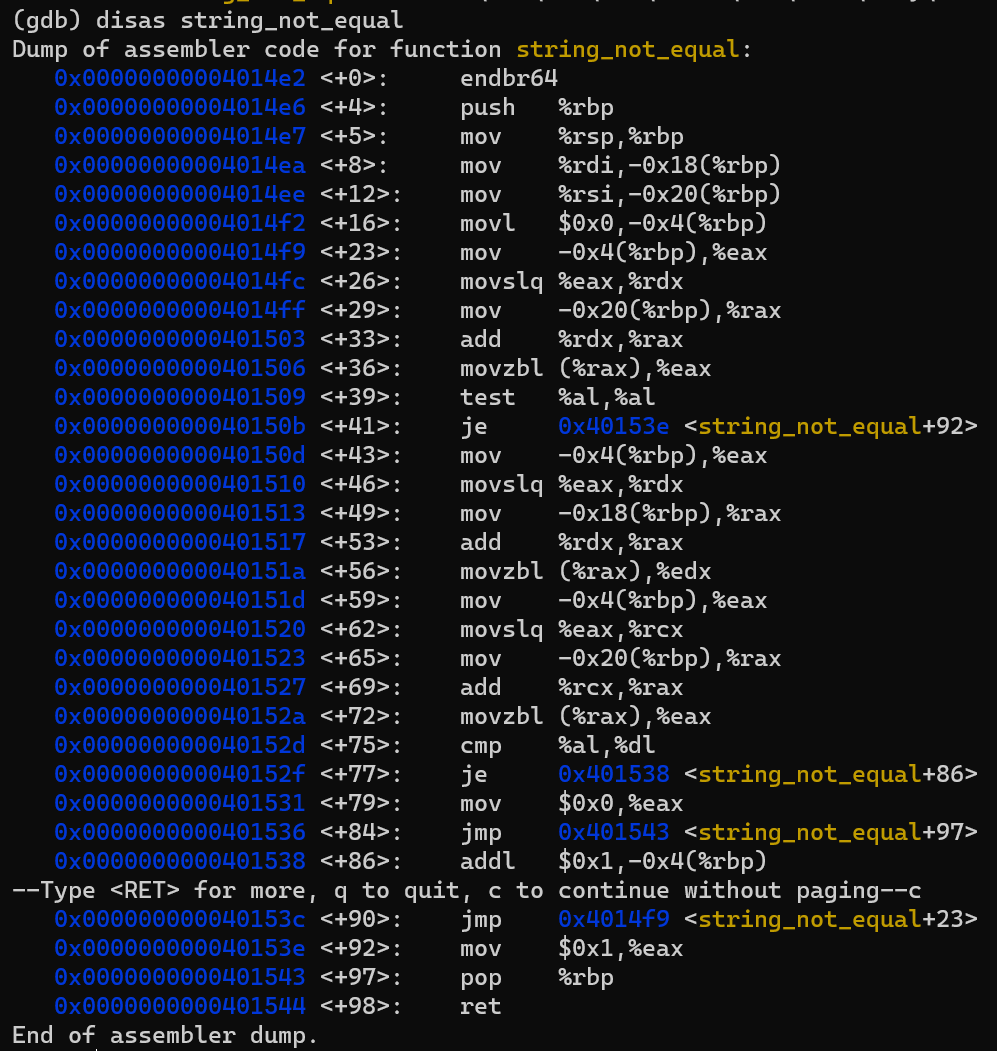
Phase\_1:



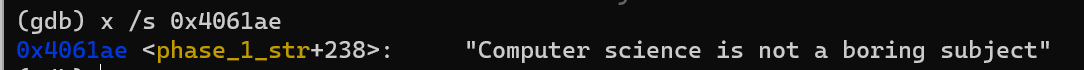
分析phase\_1调用函数string\_not\_equal的功能：将%rdi和%rsi的值写入栈中，之后获得第一个字符串中当前字符的地址，并将其加载到%al寄存器中，获得第二个字符串中当前字符的地址，并将其加载到%dl寄存器中。Cmp比较这俩寄存器中值是否相等，相等则add1，也就是比较下一个，不相等直接返回0，最终若字符串相等，将返回1（存在%eax中）。回到phase\_1：将%exa的值与1进行异或操作，也就是，如果是相等，返回1，经过异或为0。接下来测试%al寄存器的值（即%eax的低8位），是0，跳过爆炸，得到答案。



Phase\_1\_offset就存储了偏移值，我们用 x /x 语句以16进制查看它存储的偏移值

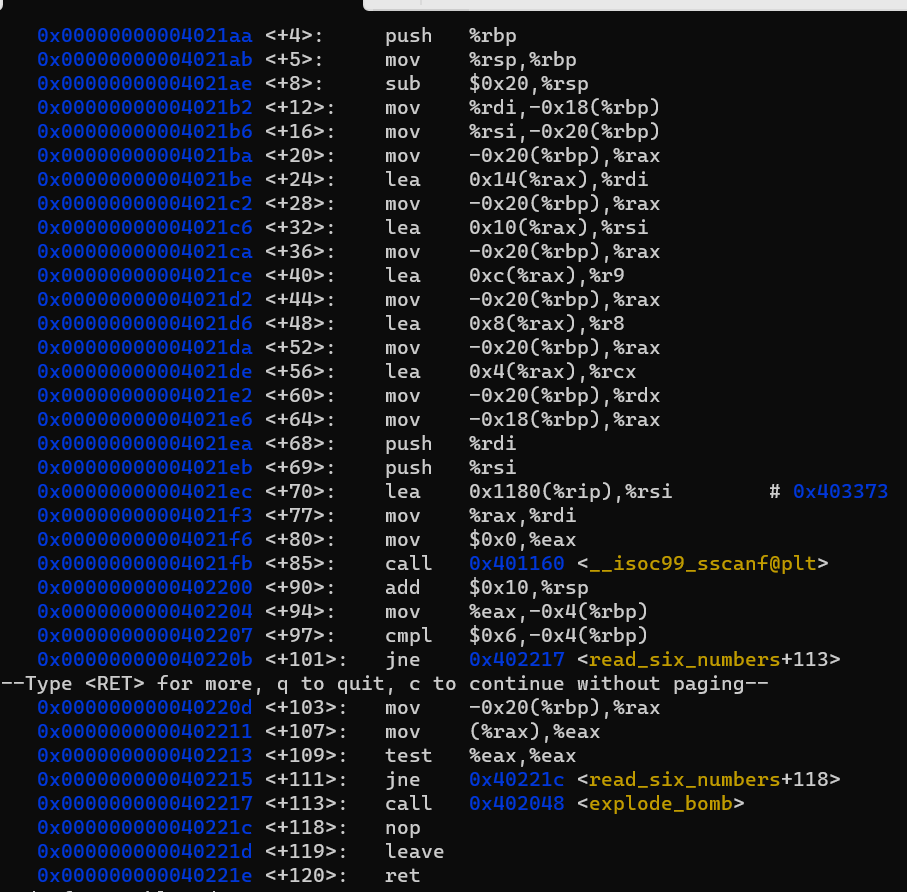


偏移量是0xee，再和phase\_1\_str的地址（4060c0）相加后得到最后地址后四位是1ae，则查询0x4061ae存储的内容

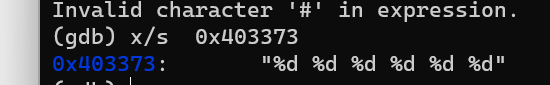


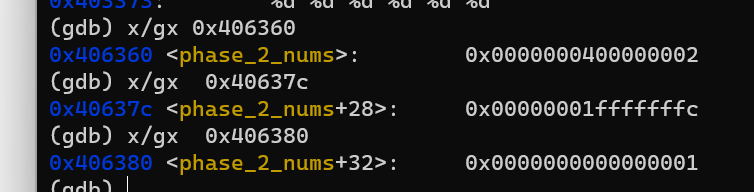
得到答案。

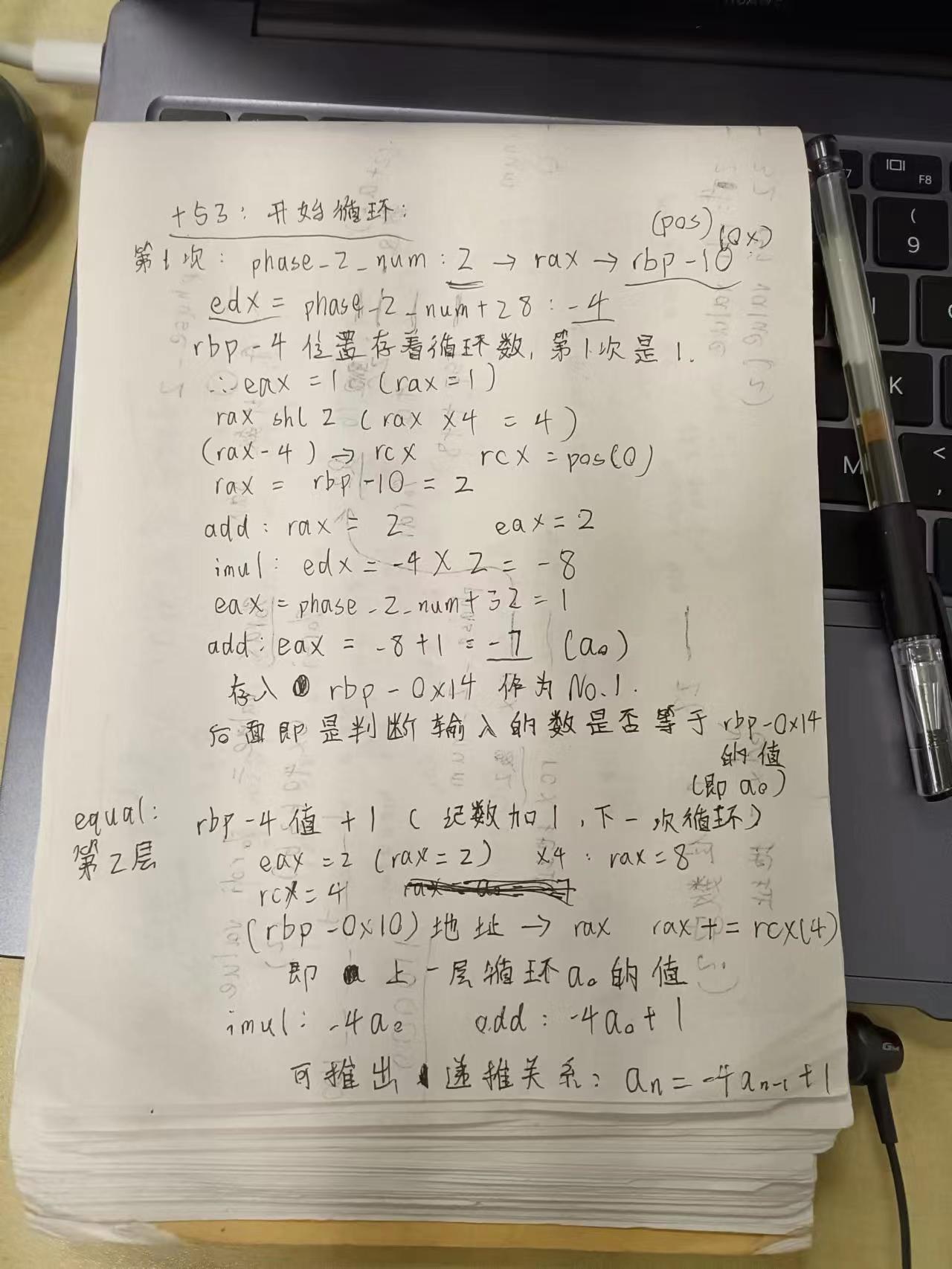
Phase\_2



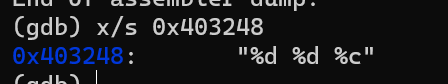
题解步骤如下，是一个循环



发现要输入六个整数。从+53那里开始循环，栈里面找了个地方记录循环次数三个关键数据大小。

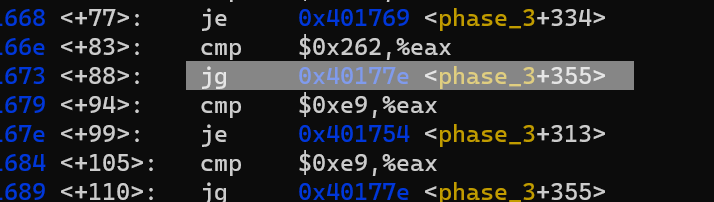


Phase\_3



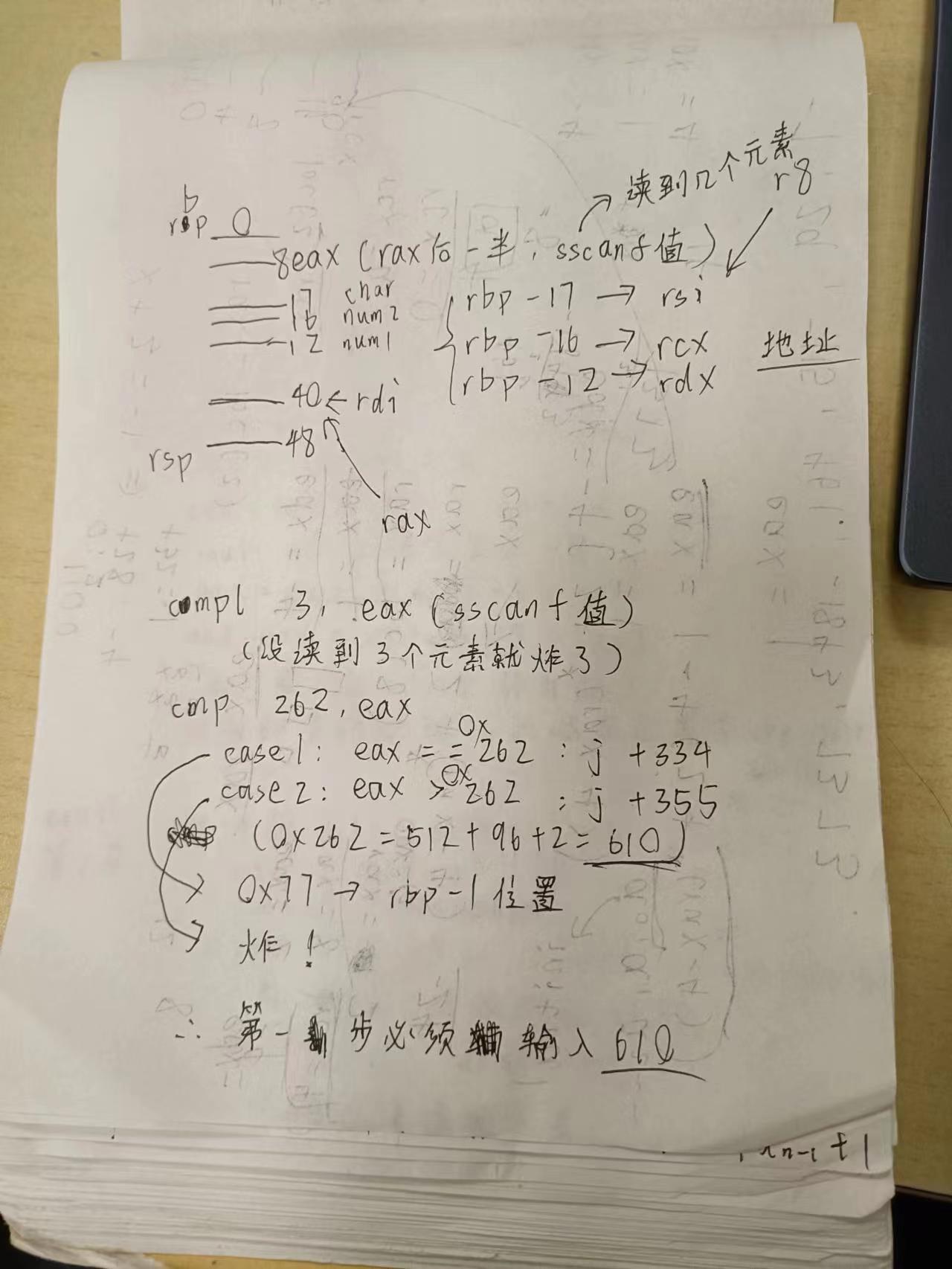
要输入“整数 整数 字符”的形式存放到rsi中。根据前面rbp-17的地址赋给rsi，rbp-16的地址赋给rcs，rbp-12的地址赋给rdx，刚好是4 4 1byte对应起来了。

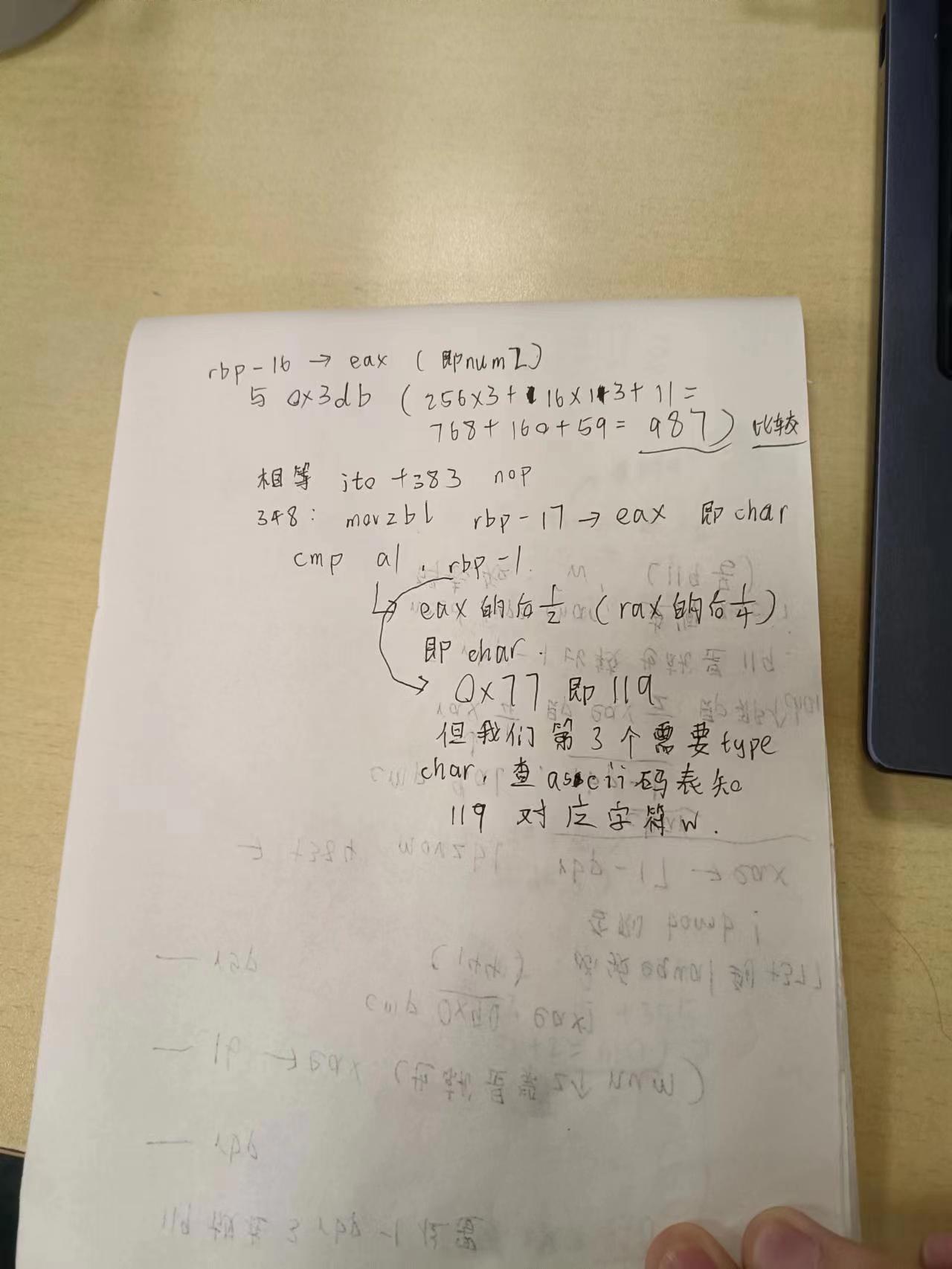
（疑惑的点在第+88行往后：但是这里先不解决，secret了再回来看



若num1小于610？没有跳转往后走了，或许是不只一个答案，也有可能是secret入口？）

分析思路如下：逐条分析跳转。

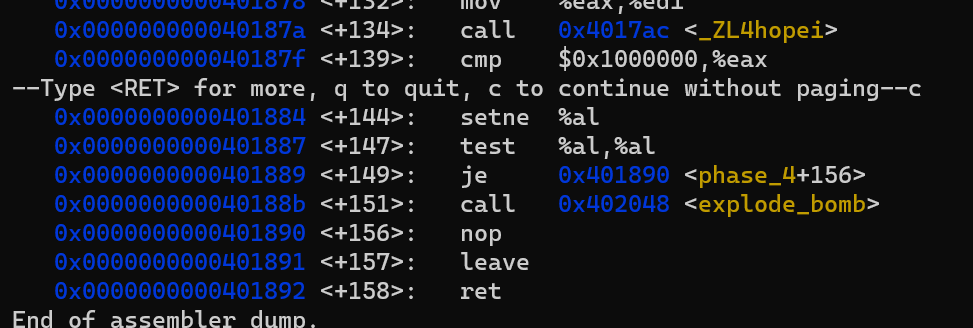




Phase\_4

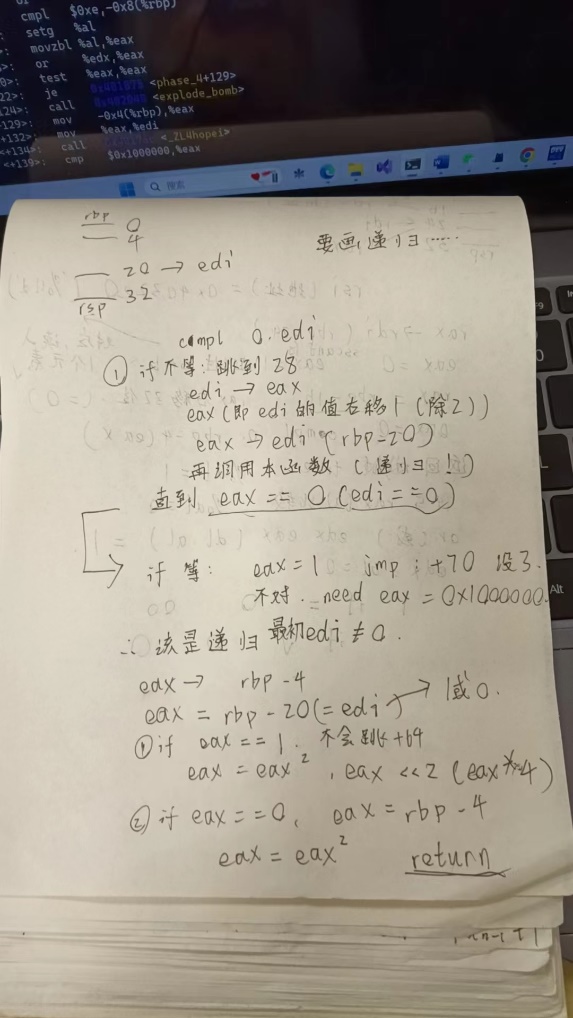
x 要输入一long long int整数

前面都是些开栈存储的操作。我先从调用函数之后逆向分析：

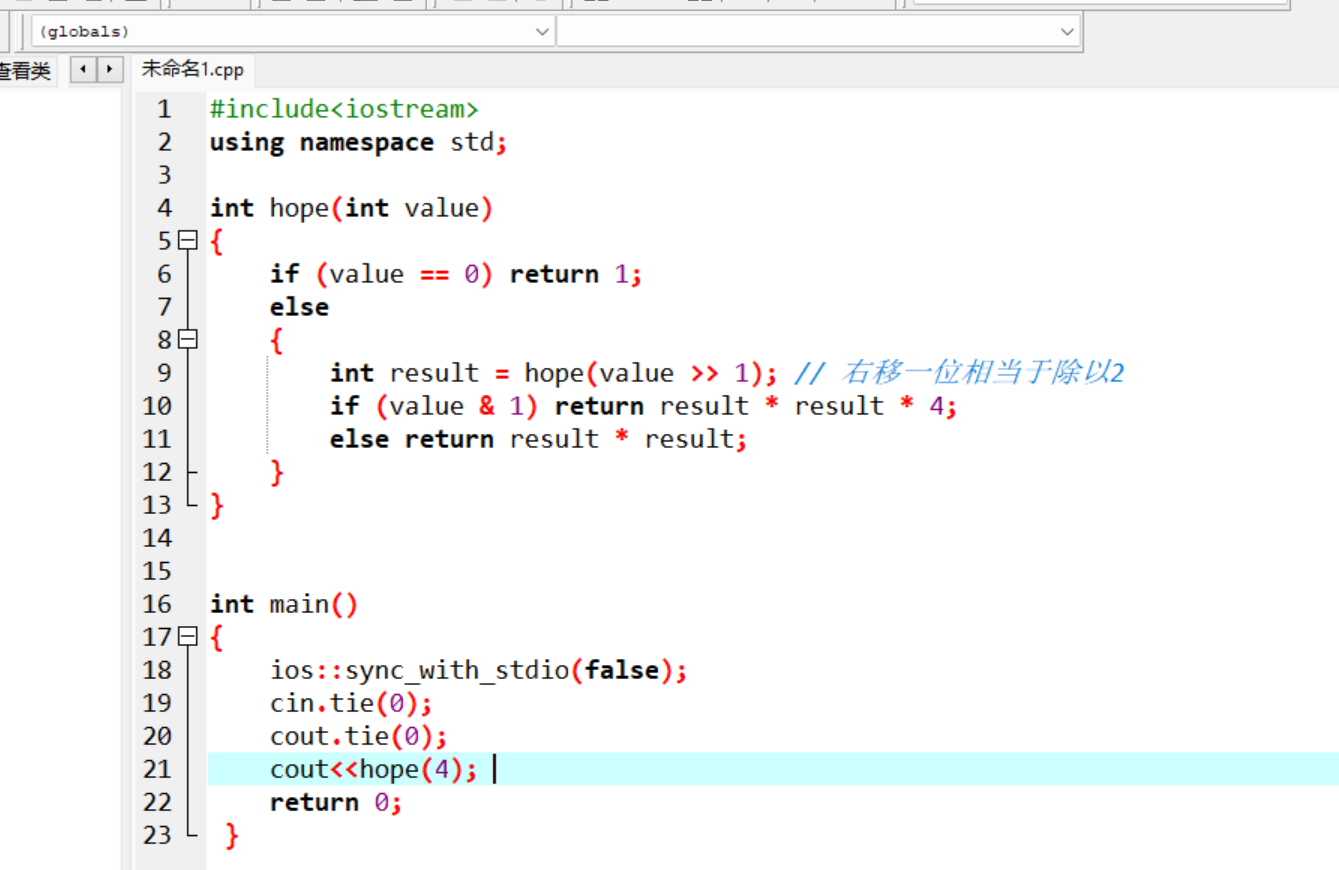


如果：比较很容易想到需要eax和0x1000000相等，就这样往下推一下：如果eax（rax，函数返回值是0x1000000，则setne（不相等则设置） al被置为0，则test 0，0（与操作）后ZF被置为1，je会跳转（跳过爆炸），这才是我们需要的。于是就可以推出，eax（函数返回值）必须是0x1000000。

接下来我们再来看这个递归函数：分析如下

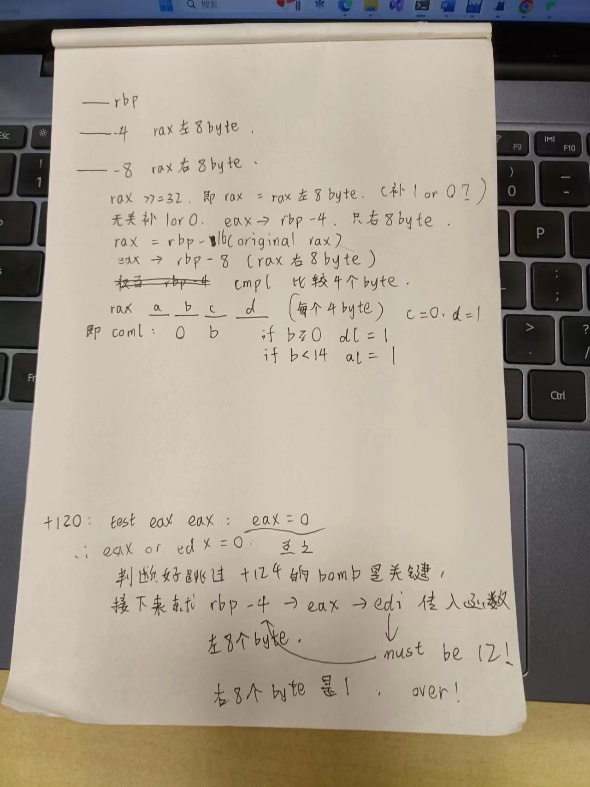
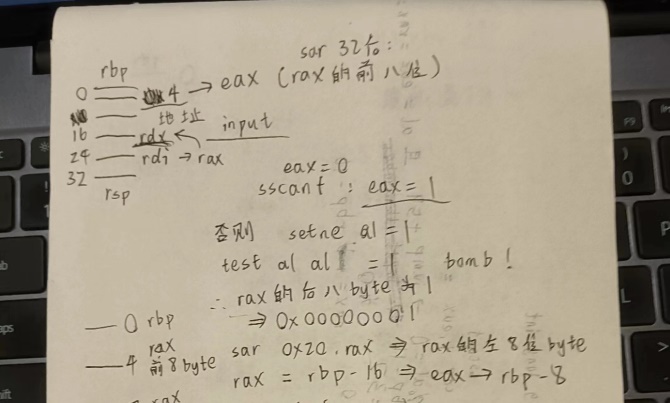


这段递归过程画起来属实麻烦，但是考虑将其转成c++语言就很好调试了。需要16的6次方，即2的24次方。模拟等价代码如下：（这是关键）



我发现输入0得到1，输入1得到4，输入4得到256，推测功能就是4的n次方，我们需要4的12次方。即，输入数该是12，即，原来寄存器edi传来的值应该是12。

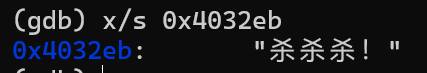
再来从phase\_4主体开始看：



最后结果：rax左八个byte是12，右八个byte是1。

Phase\_5

要输入一个字符串和一个整数

 ？？？乐

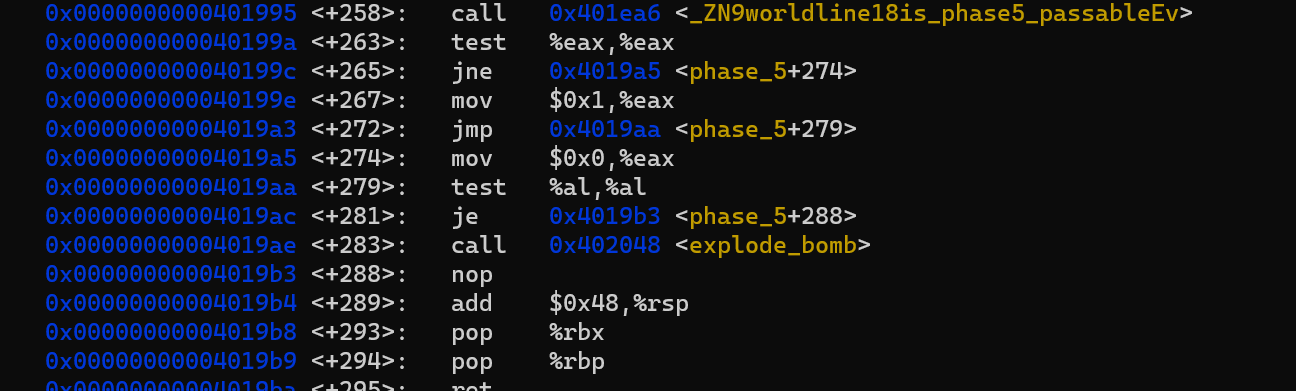
往下看，发现总共有三个地方是可以判断输入的字符串到底是啥的。先知道了在标准的C库中，strcmp函数用于比较两个字符串的内容，**如果两个字符串相等，则返回0（这一个非常重要，而不是我想当然的相等返回1）**，如果不相等则返回非零值。



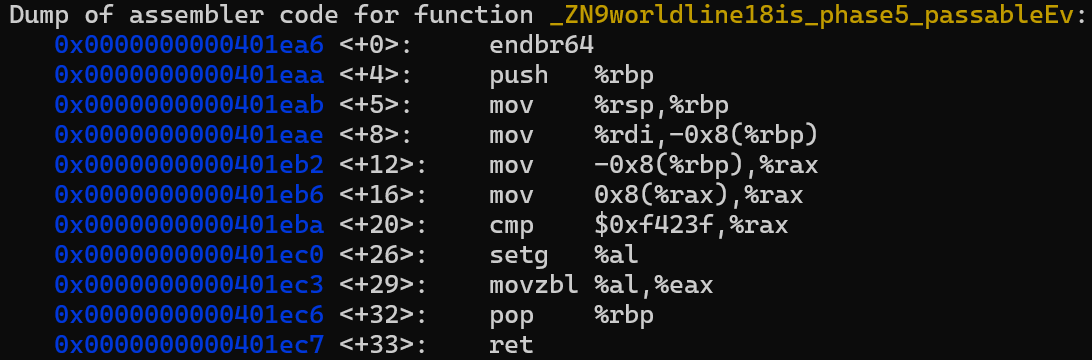
1. 如果是“杀杀杀！”返回0，不跳到114行，一直做跳到219行过了bomb，可行
2. 如果是“退退退。”78行返回1，跳到114行，128行返回0，跳到219行，可行
3. 如果是“冲冲冲~”，78—》114—》164，178行返回0，到219行，可行

都不会炸。可以断定%s是这三个中的一个。

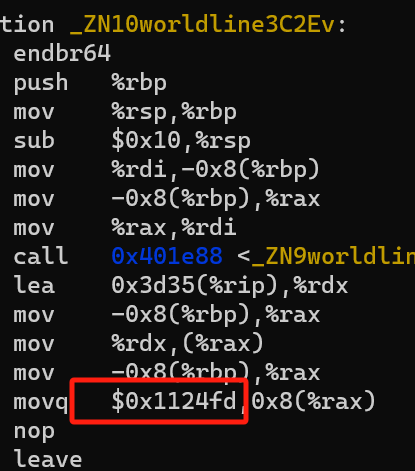
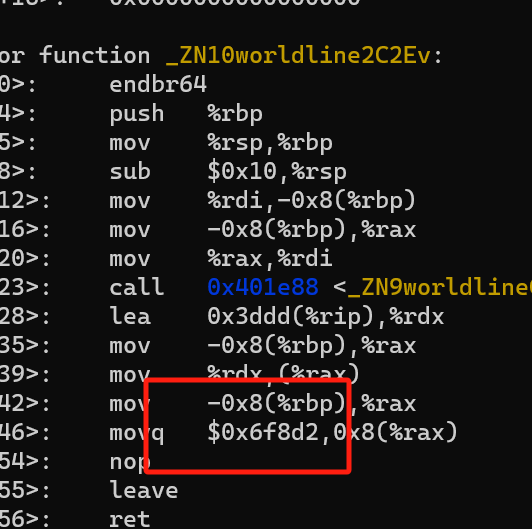
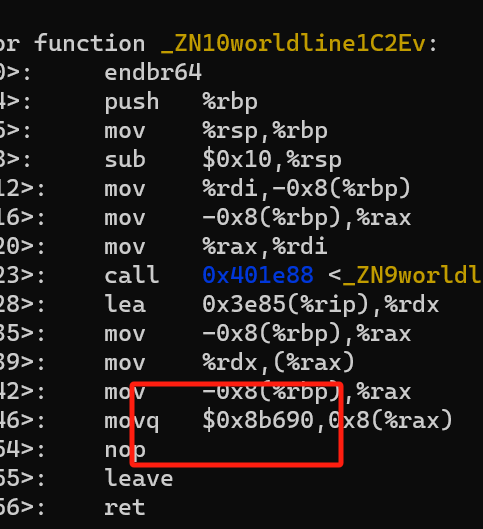
接下来判断整数是个啥。逆向倒推：



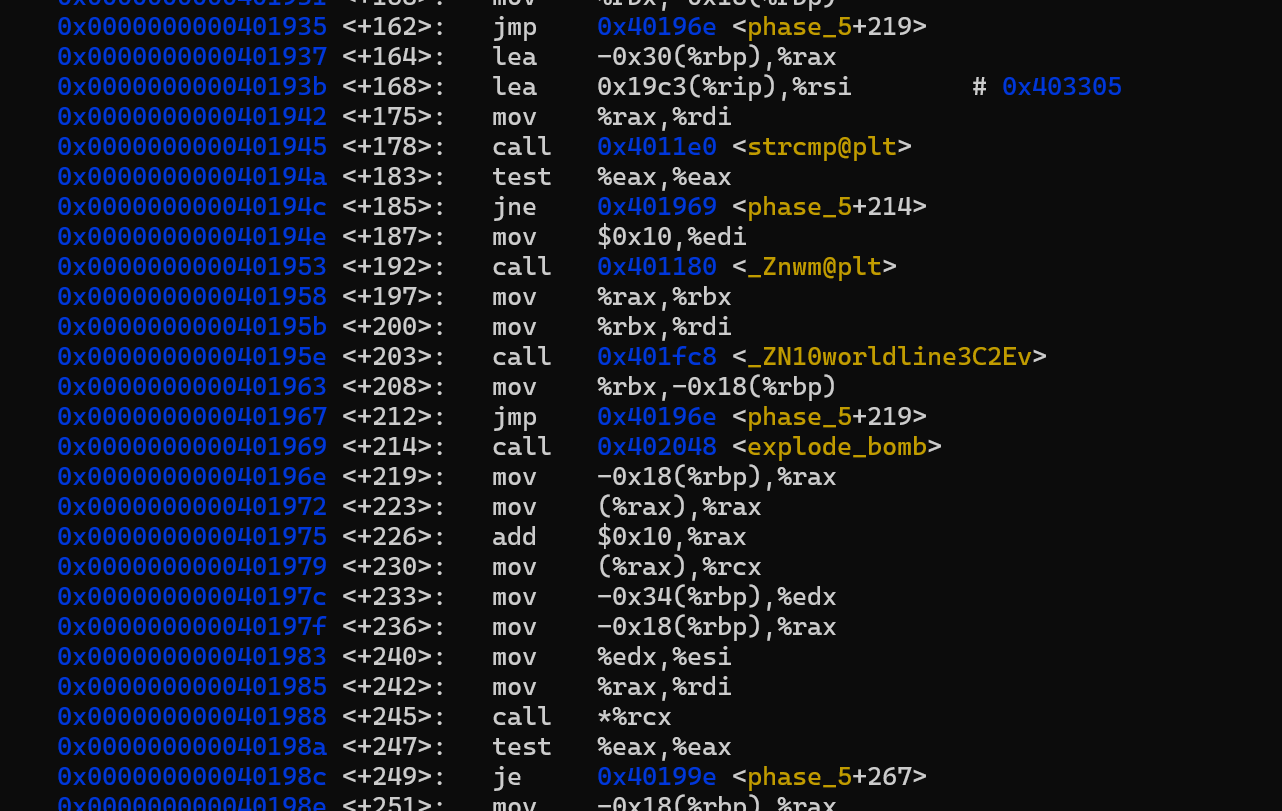
在调用0x401ea6 <\_ZN9worldline18is\_phase5\_passableEv> 之后，返回的eax要是是0，eax被赋值为1，跳到279行，test是1，不跳，bomb! 所以：函数返回的eax一定要是非0。



重点是+20那里，也就是rax（即传入参数rdi）一定要大于0xf423f，al（eax）才被设为1。

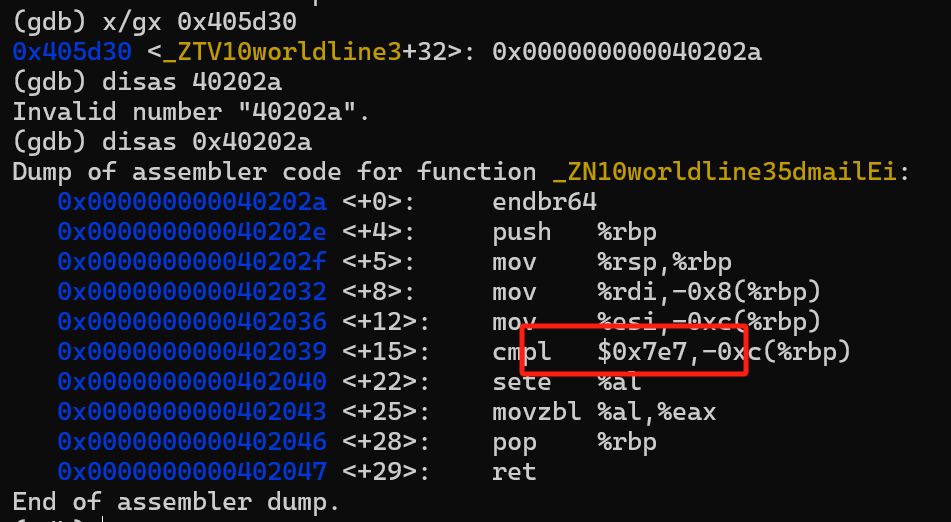


验证后发现仅有三满足条件。（“冲冲冲~”）



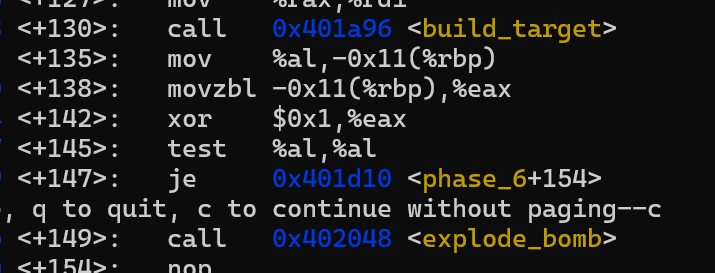
来看3的部分：比较字符串相等返回eax为0，edi赋为16，开空间（这个开空间的函数没看太懂，好像就是c++的class那种操作？反之没有对rax的值啊啥的做出任何修改，单纯只是开了个空间，不理会它）rdi=rbx=rax，调用函数ZN10，ZN10又调用了ZN9，ZN9相当于是个初始化（类似）的功能，总之ZN9修改的值传回来后ZN10又会把它修改掉。

ZN10将0x405d20这个地址的值传到（rax）里面返回了（相当于rax指向这个地址）。再往下看+223开始，这个地址指向的值赋给rax（0x405d20）并加10赋给rcx。接下来是关键：到了用rbp-0x34的值的时候了（之前完全没出现过，这地方存的就是输入的整数，也就是说终于到告诉我整数该输入什么的时候了），调用了\*rcx（rcx存的值为地址的函数）（也就是0x405d30这个地址的存的值为地址的函数（有个\*，解引用）。找到答案。



Phase\_6

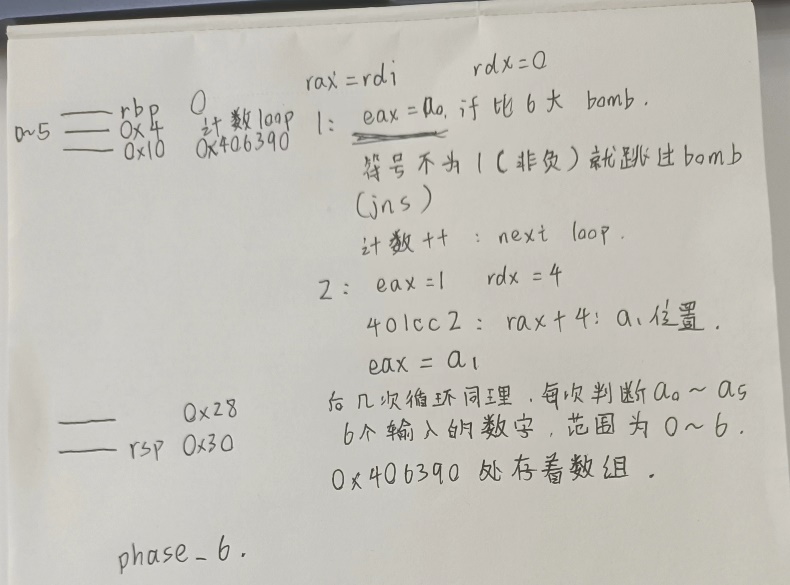
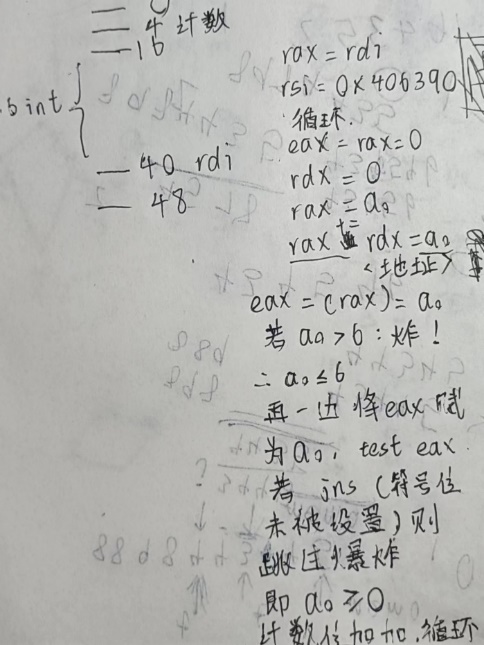
前半部分是读入六个数字，好像进行了许多判断如何不会炸，稍后再看。我还是喜欢从后往前看起：



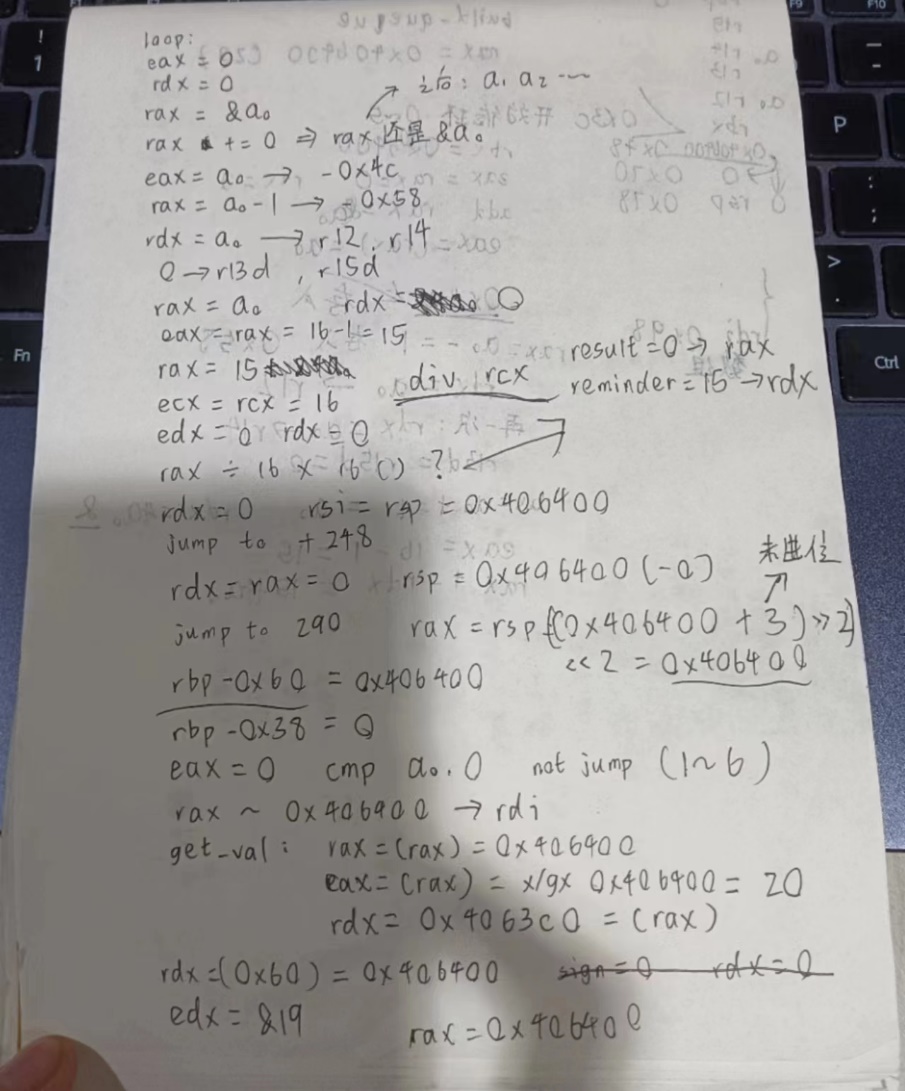
Build\_target函数返回值如果是1，eax=1，与1xor异或为0，跳过bomb，即需要该函数返回1。接下来看了看build\_targret函数：（巨长无比……）我选择再从头开始（）

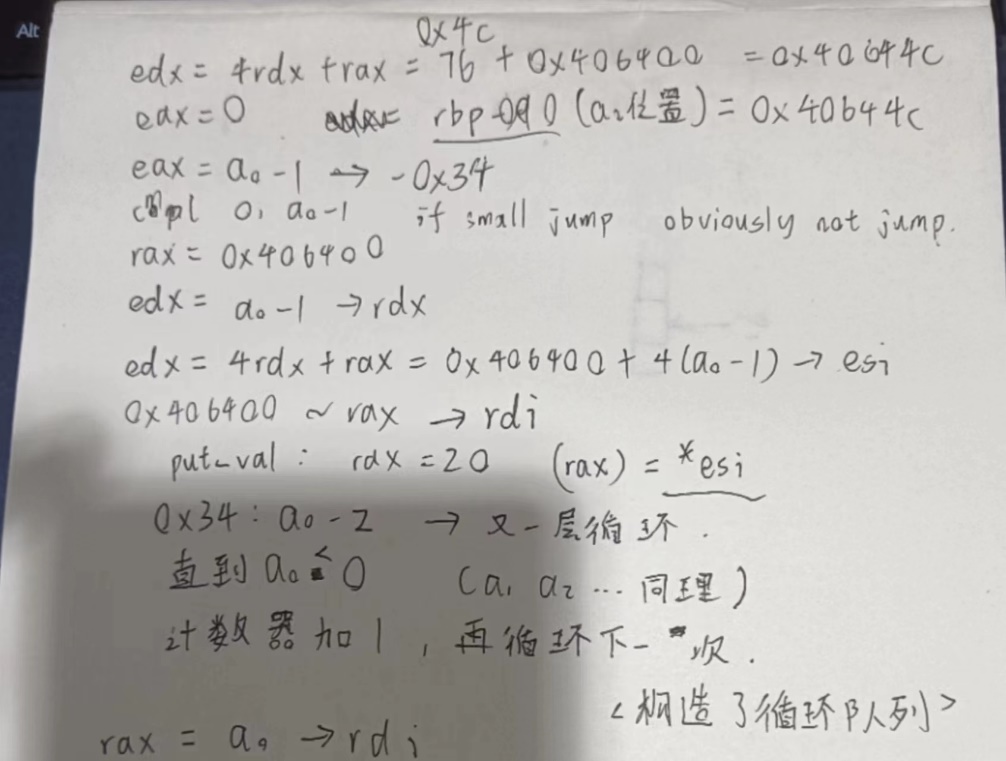
发现前面又是像第二题一样的读入六个数字，然后一个循环判断每一个数的大小范围。注意rax接受的是lea，在栈-0x10到-0x28的六个整数位置上存的是读入的六个数的地址，所以rdx每次是0，4，8……加到rax上，相当于是移到数组的下一个数字来判断，分析结果如下图所示：

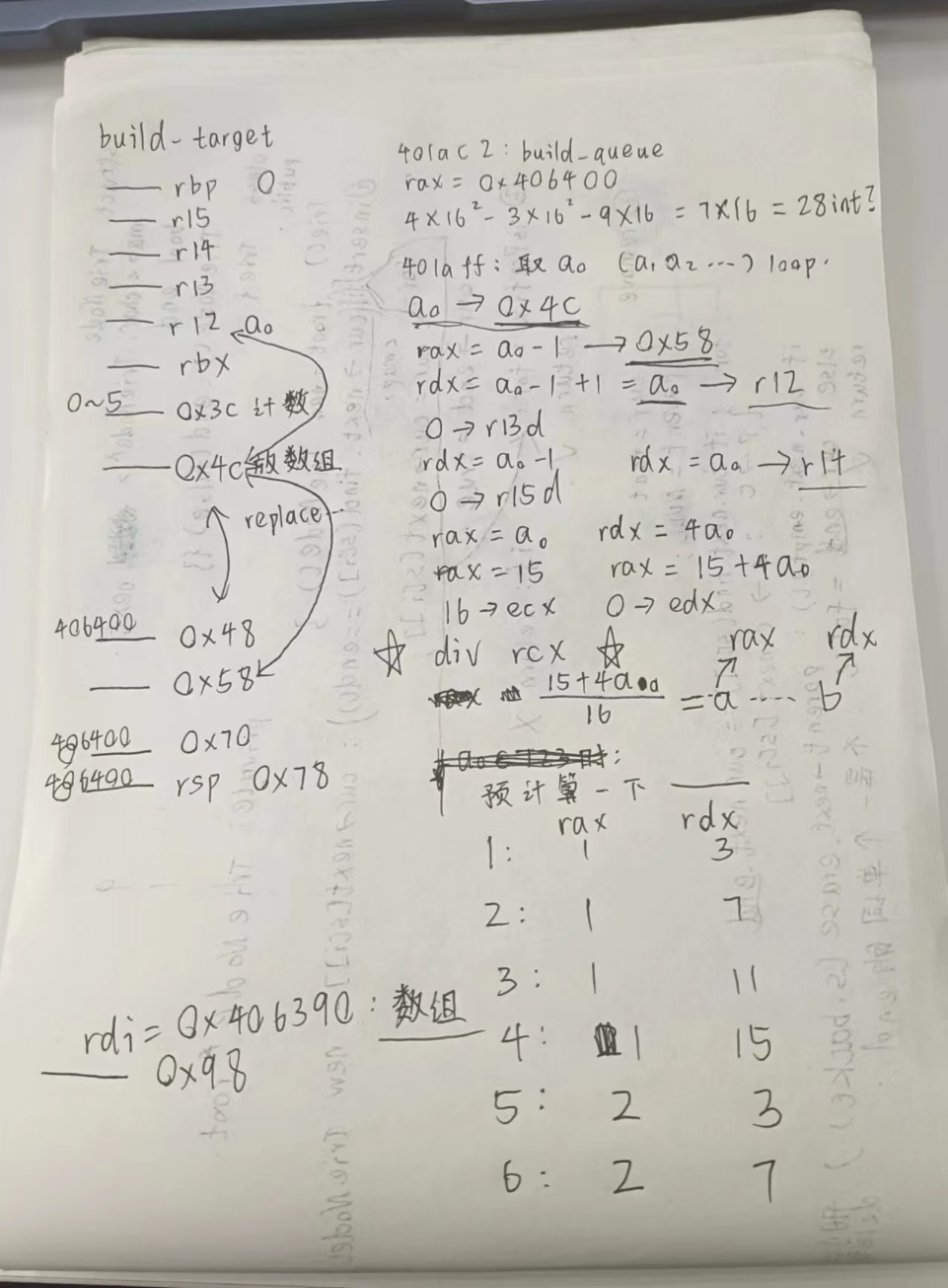
得到输入的数字必须为1~6这六个数字。（key）

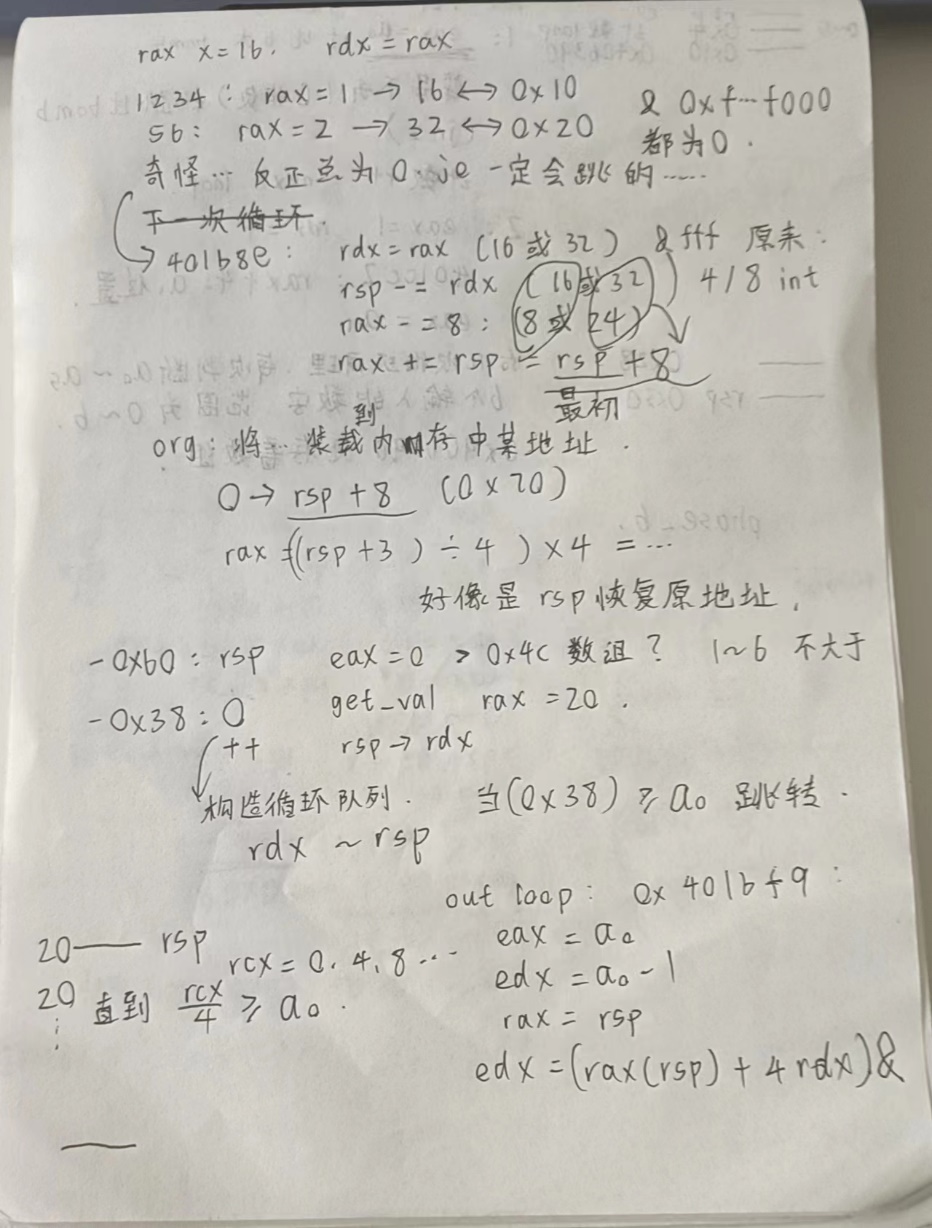


接下来分析build\_target：（基本只过了一遍，过了第一次循环）



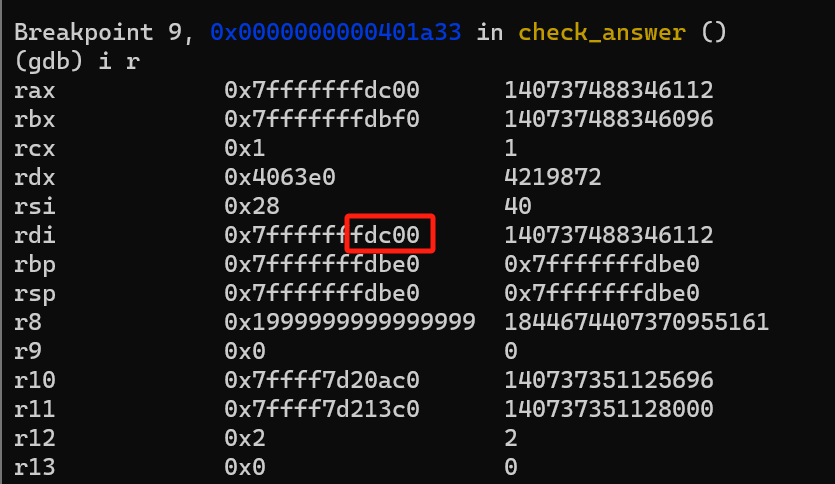






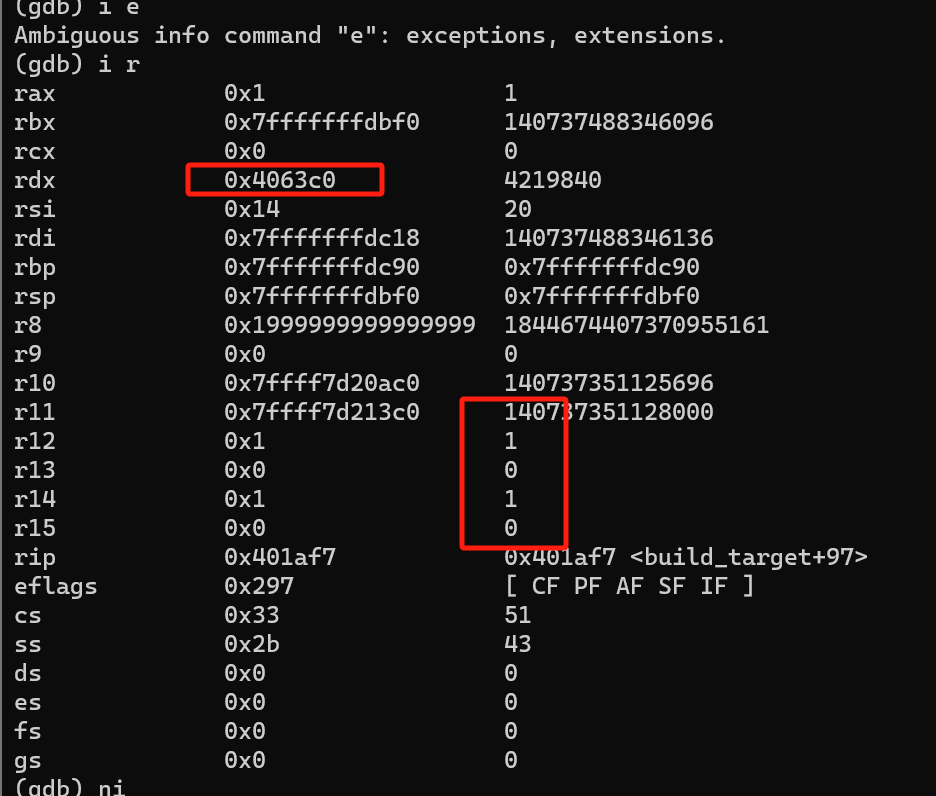
在build\_target中有一个非常重要的是div rcx，这句话是把rax/rcx，计算得到的值存入rax并且余数存入rdx。

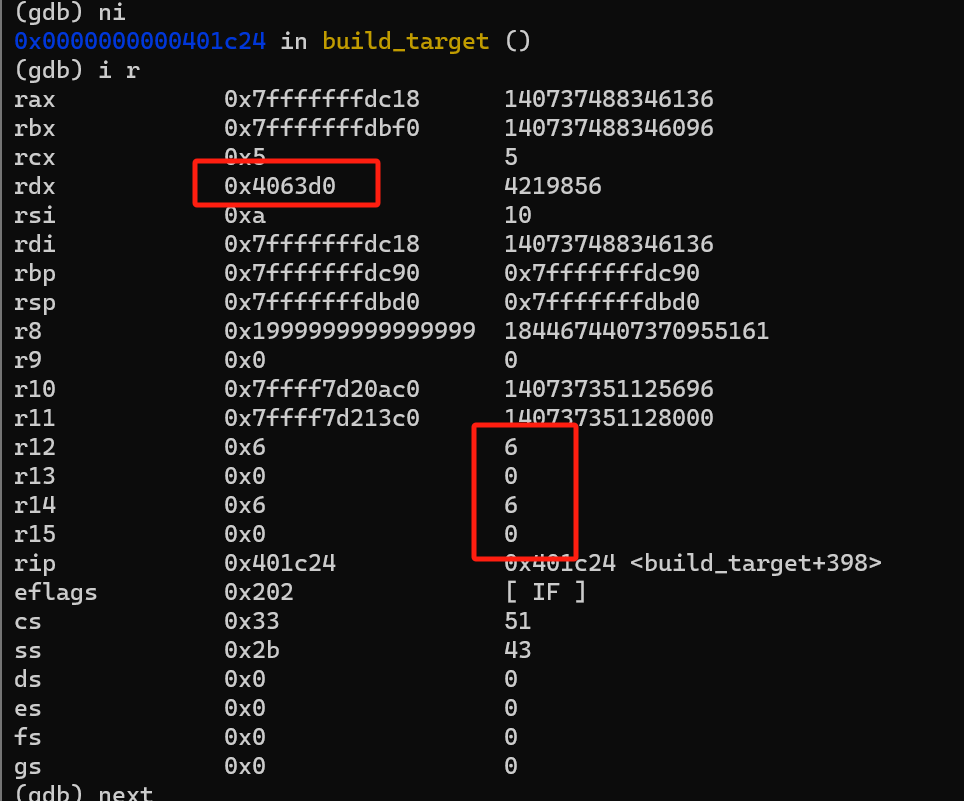
r12和r14在每一次循环中都储存着数组a0—a6，rcx则存储着a0-1,a1-1……a6-1

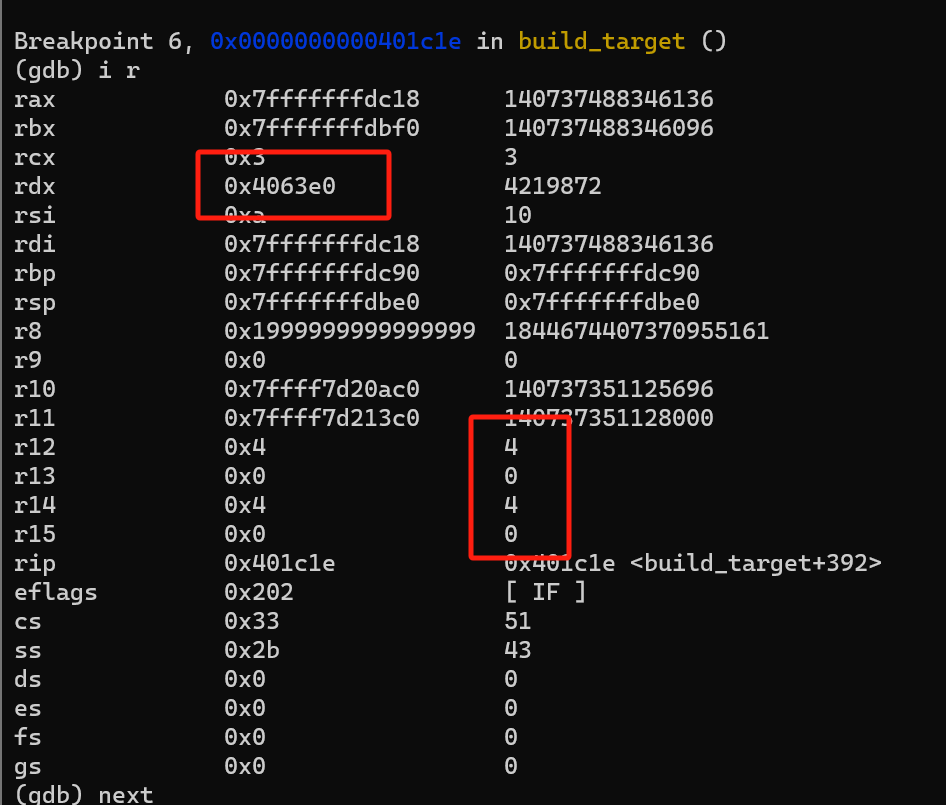


最后check\_answer是将dc00处的首地址传进来的。

对应前面在4063c0—406410处的六个值即可最终找到答案：

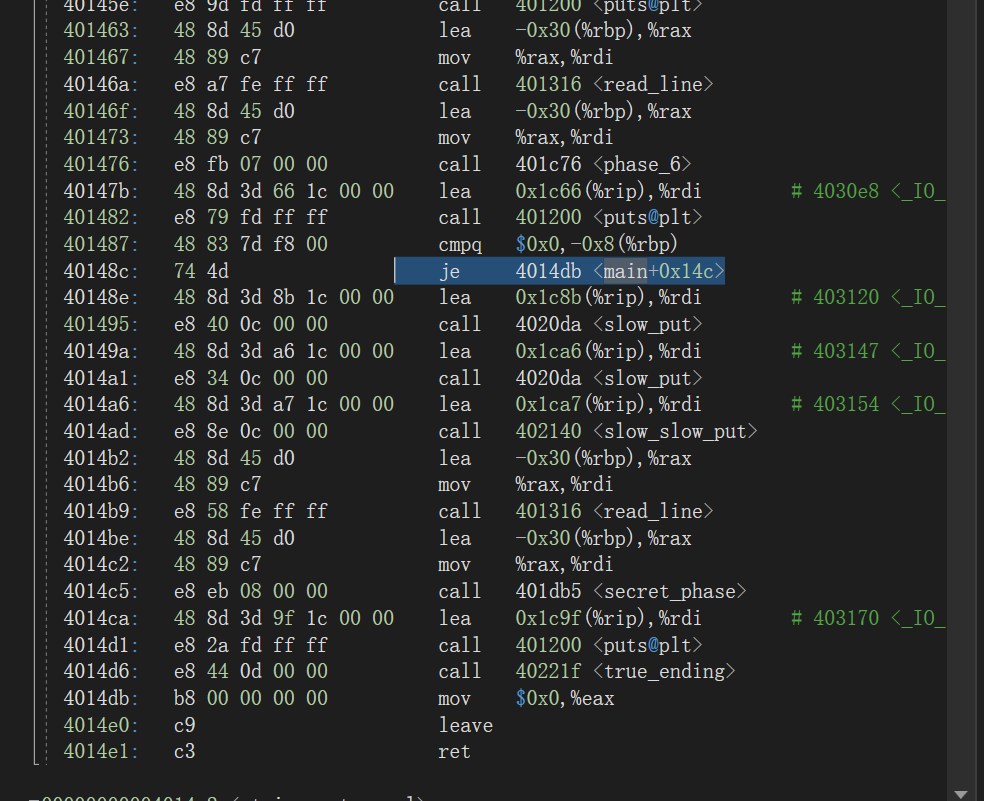






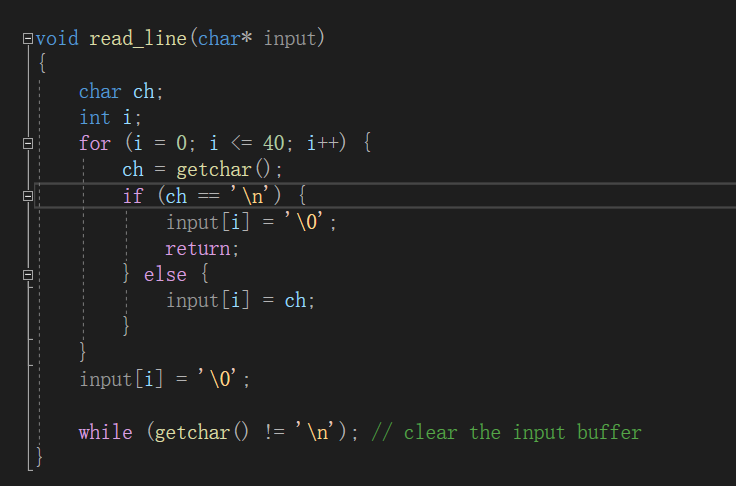
Secret\_phase:

在main函数中，我们看到这一段：在je这里它跳过了secret

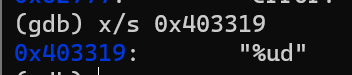


也就是说，想要进入这里，得让rbp-8的位置存储的数不等于0即可。

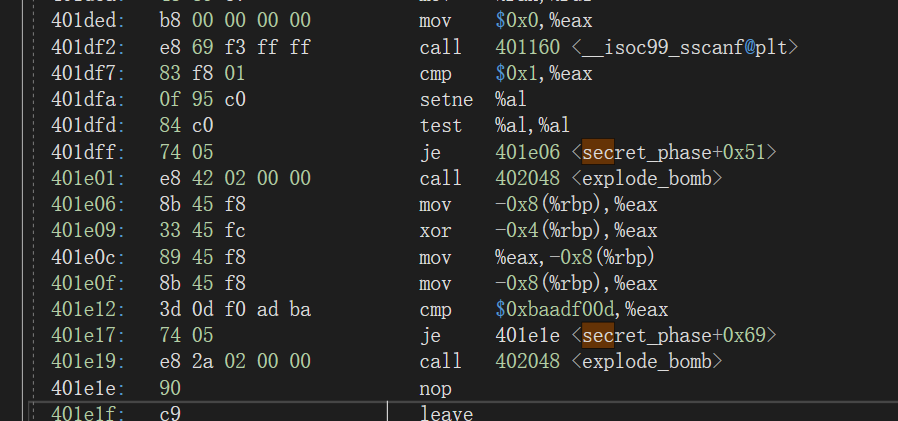
“你或许能从main.cpp中发现某些违和之处……”



在read\_line中，字符结束后只要不是换行都还能继续读入，而且注意到在bomb中只要不是有新的%s%d%c啥的也不会被切割。然后，在最后做完六题后我打了个换行，粘贴到run里面就炸了，于是就可以想到，读入的字符，结尾某些元素是不能加的某些元素是可以加的。新元素不能加，但空格不会被sscanf切割，于是在答案后打空格，果然没炸，进入了secret。

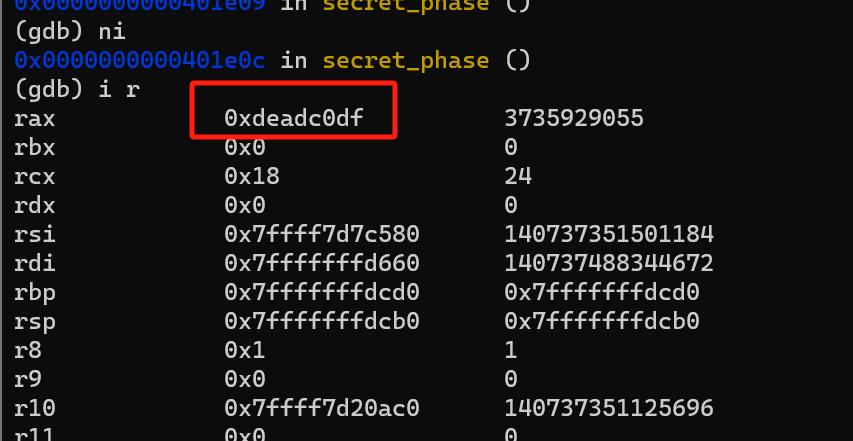


要求输入一个无符号整数。

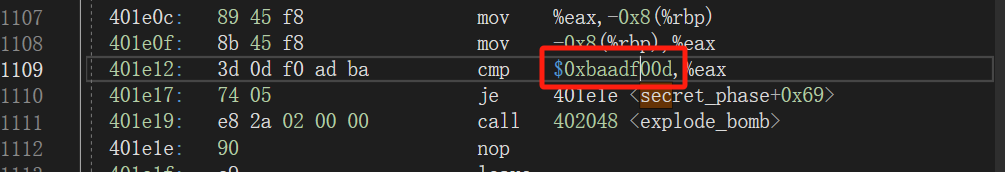


后半部分：cmp那里eax必须为1，al才为0，je跳过爆炸，也就是要输入一个东西，对应前面，就是要输入一个无符号整数。先随便输个数去gdb调试：

Eax就是我随便输入的数字1在rbp-8的位置，和rbp-4位置的元素异或后值为



也就是，rbp-8位置的数是：0xdeadcde。输入与这个数异或后的值得是这

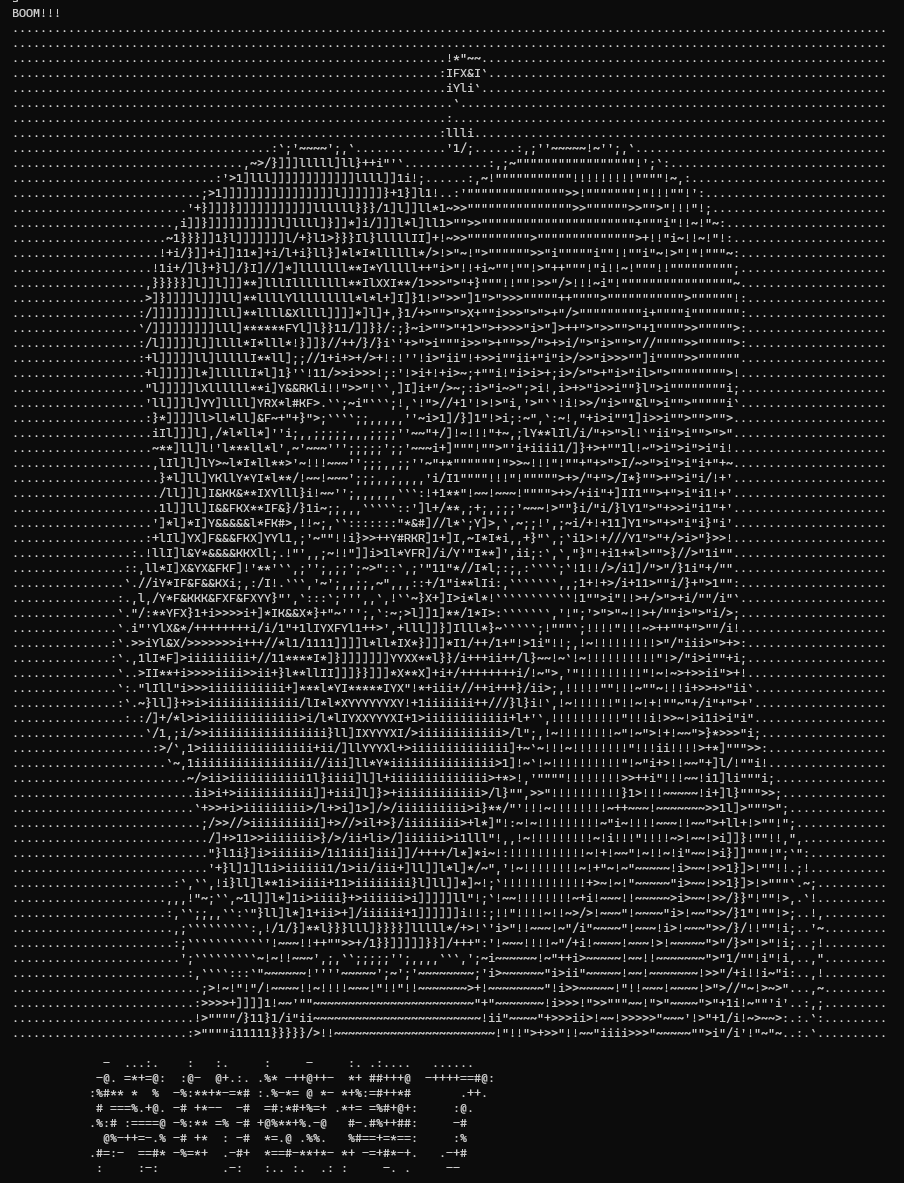


x—deadc0de

y—baadf00d

input—640030d3

转为十进制，即1677734099。完成！！！！！！！！！！！！！！



|可爱捏波喜多

|

|

|

|  
|

|

|

|

