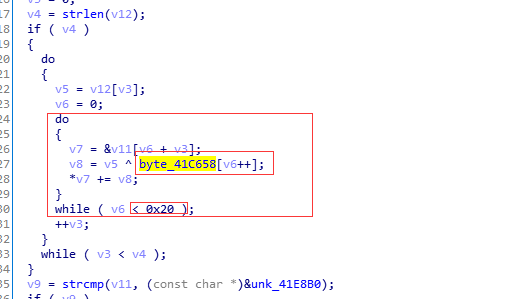
# 题目分析

我们拖进IDA可以发现，程序的逻辑是十分清晰的，进行计算的部分主要就是两个do-while循环，但是我们可以发现在第二个do-while循环里面，使用到了byte\_41C658数组中的每一个数据了的，因此，如果进行逆向计算就会十分困难，我们选择用Z3库来解。



from z3 import \*  
s = Solver()  
flag = [BitVec(('x%d' % i),8) for i in range(0x22) ]  
data1 = [0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,0x3A,  
0x3B,0x3C,0x3D,0x3E,0x3F,0x40,0x5B,0x5C,0x5D,0x5E,0x5F,0x60,0x7B,0x7C,0x7D,0x7E]  
data2 = [0x72,0xE9,0x4D,0xAC,0xC1,0xD0,0x24,0x6B,0xB2,0xF5,0xFD,0x45,0x49,0x94,0xDC,0x10,  
0x10,0x6B,0xA3,0xFB,0x5C,0x13,0x17,0xE4,0x67,0xFE,0x72,0xA1,0xC7,0x04,0x2B,0xC2,  
0x9D,0x3F,0xA7,0x6C,0xE7,0xD0,0x90,0x71,0x36,0xB3,0xAB,0x67,0xBF,0x60,0x30,0x3E,  
0x78,0xCD,0x6D,0x35,0xC8,0x55,0xFF,0xC0,0x95,0x62,0xE6,0xBB,0x57,0x34,0x29,0x0E,3]  
xor\_result = [0]\*0x41  
for m in range(0,0x22):  
 for n in range(0,0x20):  
 xor\_result[n+m] += flag[m] ^ data1[n]  
for o in range(0,0x41):  
 s.add(xor\_result[o] == data2[o])  
  
if s.check() == sat:  
 model = s.model()  
 str = []  
 #str = [chr(model[flag[i]].as\_long().real) for i in range(36)]  
 for i in range(0x22):  
 str += chr(model[flag[i]].as\_long().real)  
 print "".join(str)  
else:  
 print "unsat"

flag：SYC{4+mile+b3gin+with+sing1e+step}