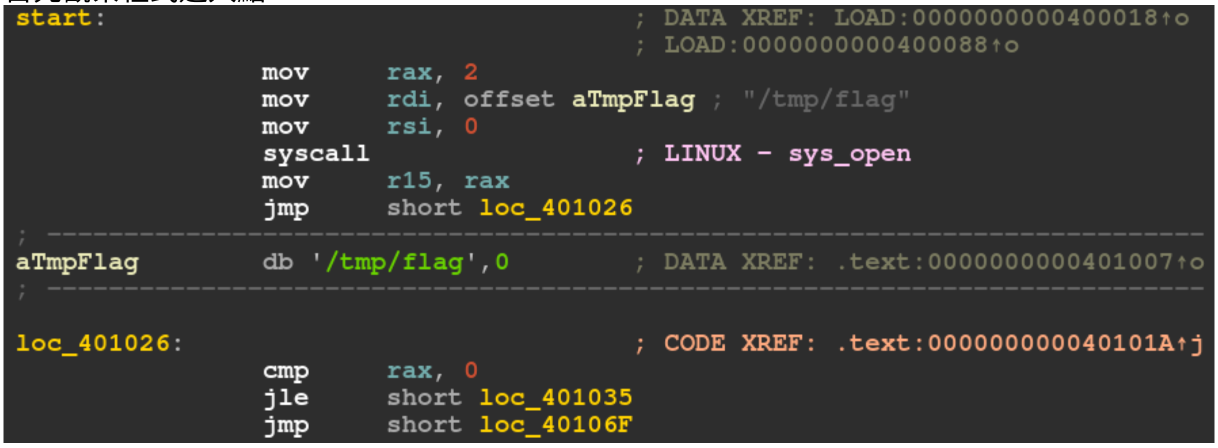
sacred arts

題目是一個執行檔，使用IDA反組譯題目。

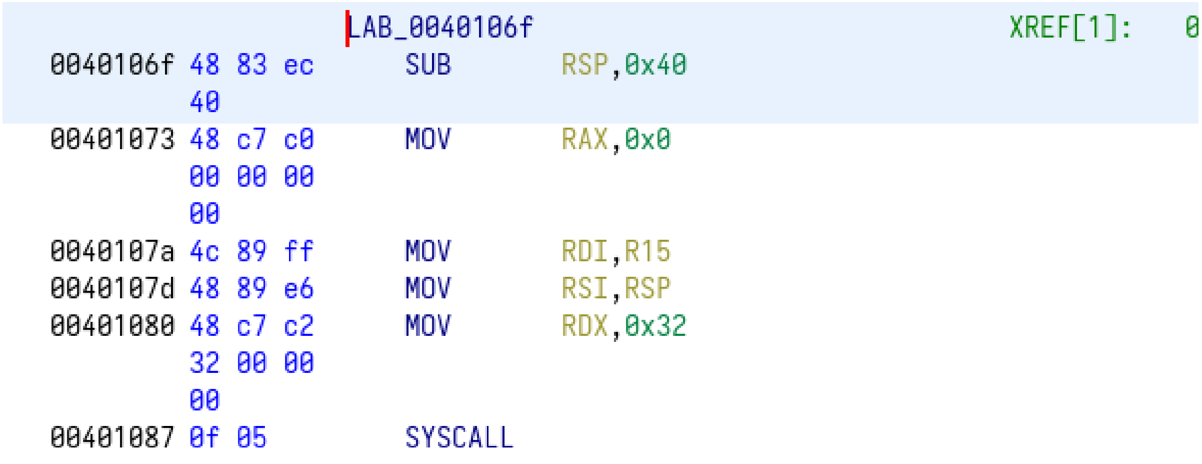
start

首先觀察程式進入點



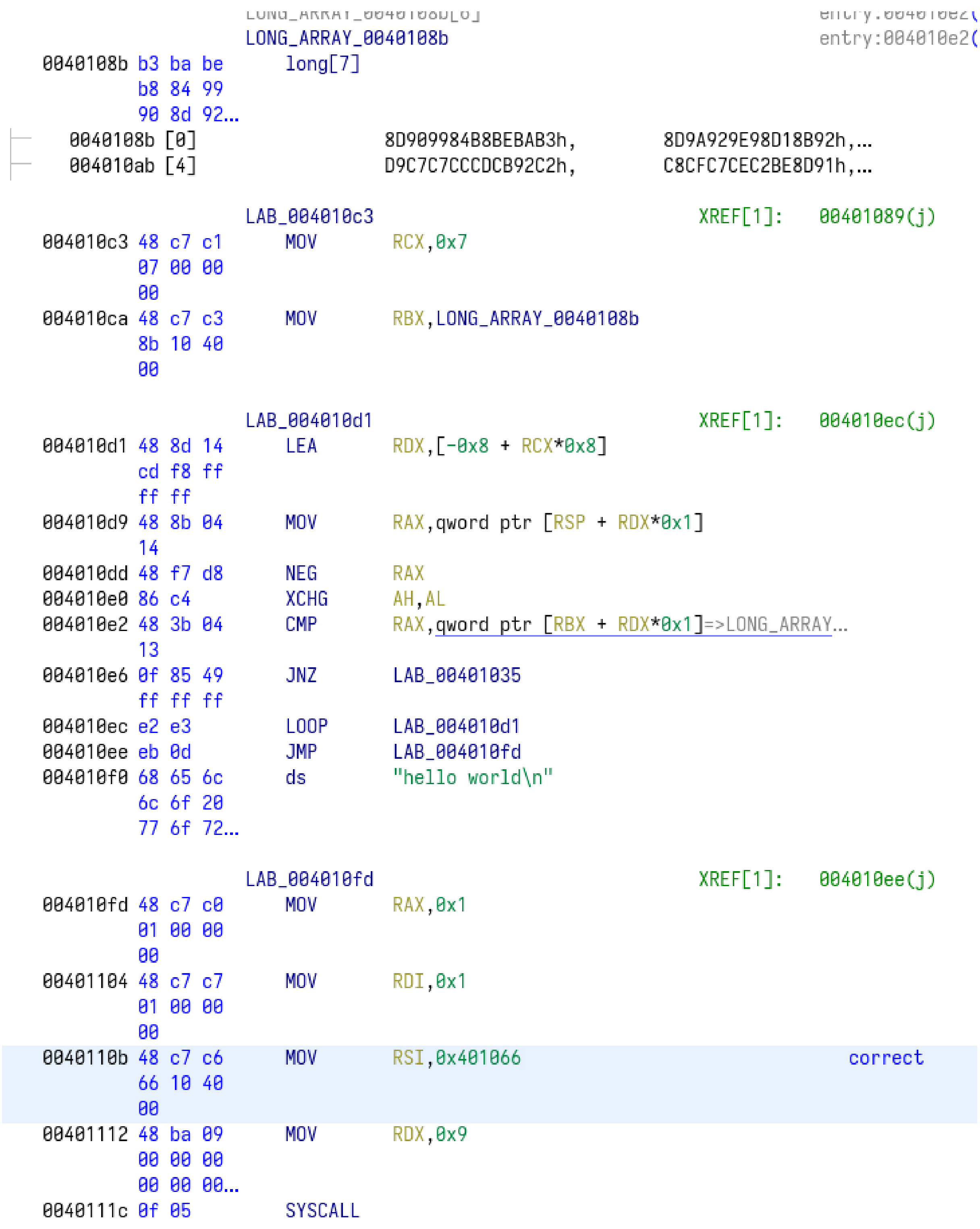
```
start:                                  ; DATA XREF: LOAD:0000000000400018↑o
                                        ; LOAD:0000000000400088↑o
                mov     rax, 2
                mov     rdi, offset aTmpFlag ; "/tmp/flag"
                mov     rsi, 0
                syscall                 ; LINUX - sys_open
                mov     r15, rax
                jmp     short loc_401026
; ---------------------------------------------------------------------------
aTmpFlag        db '/tmp/flag',0        ; DATA XREF: .text:0000000000401007↑o
; ---------------------------------------------------------------------------

loc_401026:                             ; CODE XREF: .text:000000000040101A↑j
                cmp     rax, 0
                jle     short loc_401035
                jmp     short loc_40106F
```

首先可以看到在start一開始的地方做了一個syscall，
根據 `mov rax, 2` 可以知道這是 syscall open，
以及根據 `mov rdi, (offset) aTmpFlag` 可以得知路徑爲 “/tmp/flag”
做完syscall open後，他會做syscall read，讀長度爲0x32的資料



```
                          LAB_0040106f                                XREF[1]:     0
     0040106f 48 83 ec       SUB        RSP,0x40
              40
     00401073 48 c7 c0       MOV        RAX,0x0
              00 00 00
              00
     0040107a 4c 89 ff       MOV        RDI,R15
     0040107d 48 89 e6       MOV        RSI,RSP
     00401080 48 c7 c2       MOV        RDX,0x32
              32 00 00
              00
     00401087 0f 05          SYSCALL
```

繼續往下走，結果就看到了核心邏輯:



```
                        LONG_ARRAY_0040108b[0]                            entry:004010e2(
                        LONG_ARRAY_0040108b                               entry:004010e2(
    0040108b b3 ba be      long[7]
             b8 84 99
             90 8d 92...
      0040108b [0]                 8D909984B8BEBAB3h,        8D9A929E98D18B92h,...
      004010ab [4]                 D9C7C7CCCDCB92C2h,        C8CFC7CEC2BE8D91h,...

                        LAB_004010c3                          XREF[1]:    00401089(j)
    004010c3 48 c7 c1      MOV        RCX,0x7
             07 00 00
             00
    004010ca 48 c7 c3      MOV        RBX,LONG_ARRAY_0040108b
             8b 10 40
             00

                        LAB_004010d1                          XREF[1]:    004010ec(j)
    004010d1 48 8d 14      LEA        RDX,[-0x8 + RCX*0x8]
             cd f8 ff
             ff ff
    004010d9 48 8b 04      MOV        RAX,qword ptr [RSP + RDX*0x1]
             14
    004010dd 48 f7 d8      NEG        RAX
    004010e0 86 c4         XCHG       AH,AL
    004010e2 48 3b 04      CMP        RAX,qword ptr [RBX + RDX*0x1]=>LONG_ARRAY...
             13
    004010e6 0f 85 49      JNZ        LAB_00401035
             ff ff ff
    004010ec e2 e3         LOOP       LAB_004010d1
    004010ee eb 0d         JMP        LAB_004010fd
    004010f0 68 65 6c      ds         "hello world\n"
             6c 6f 20
             77 6f 72...

                        LAB_004010fd                          XREF[1]:    004010ee(j)
    004010fd 48 c7 c0      MOV        RAX,0x1
             01 00 00
             00
    00401104 48 c7 c7      MOV        RDI,0x1
             01 00 00
             00
    0040110b 48 c7 c6      MOV        RSI,0x401066                                correct
             66 10 40
             00
    00401112 48 ba 09      MOV        RDX,0x9
             00 00 00
             00 00 00...
    0040111c 0f 05         SYSCALL
```

這邊ghidra沒能反編譯出來，但根據我自己的理解，反編譯成下面的pseudo code:

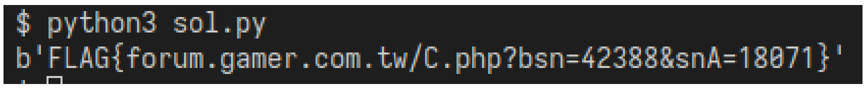
```c
uint64_t arr[] = {0x8D909984B8BEBAB3, 0x8D9A929E98D18B92,
                  0xD0888BD19290D29C, 0x8C9DC08F978FBDD1,
                  0xD9C7C7CCCDCB92C2, 0xC8CFC7CEC2BE8D91,
                  0xFFFFFFFFFFFFCF82};

scanf("%s", &flag);
for (int i = 7; i != 0; i--) {
    uint64_t rax = arr[i];
    rax = -rax;
    __asm__("xchg ah, al");
    if (rax != flag[i]) {
        puts("wrong");
        return;
    }
}
puts("correct");
```

Solve

根據上面的pseudo code，寫程式去還原flag

```python
from Crypto.Util.number import long_to_bytes

long_arr = [0x8D909984B8BEBAB3, 0x8D9A929E98D18B92,
            0xD0888BD19290D29C, 0x8C9DC08F978FBDD1,
            0xD9C7C7CCCDCB92C2, 0xC8CFC7CEC2BE8D91,
            0xFFFFFFFFFFFFCF82]

flag = b""
for rax in long_arr:
    # xchg al, ah
    al = rax & 0xFF    # extract al (7-0 bit)
    ah = (rax & 0xFF00) >> 8  # extract ah (15-8 bit)

    rax &= 0xFFFFFFFFFFFF0000  # clear al and ah

    al, ah = ah, al  # xchg
    rax |= al  # set al
    rax |= (ah << 8)  # set ah
    # neg rax
    rax = rax ^ 0xFFFFFFFFFFFFFFFF

    flag += long_to_bytes(rax)[::-1]  # little endian

print(flag)
```

Result



```
$ python3 sol.py
b'FLAG{forum.gamer.com.tw/C.php?bsn=42388&snA=18071}'
```