2021/05/26_x64汇编与逆向_第3课_流程控制语句的识别

笔记本： x64汇编与逆向
创建时间： 2021/5/26 星期三 10:05
作者： ileemi

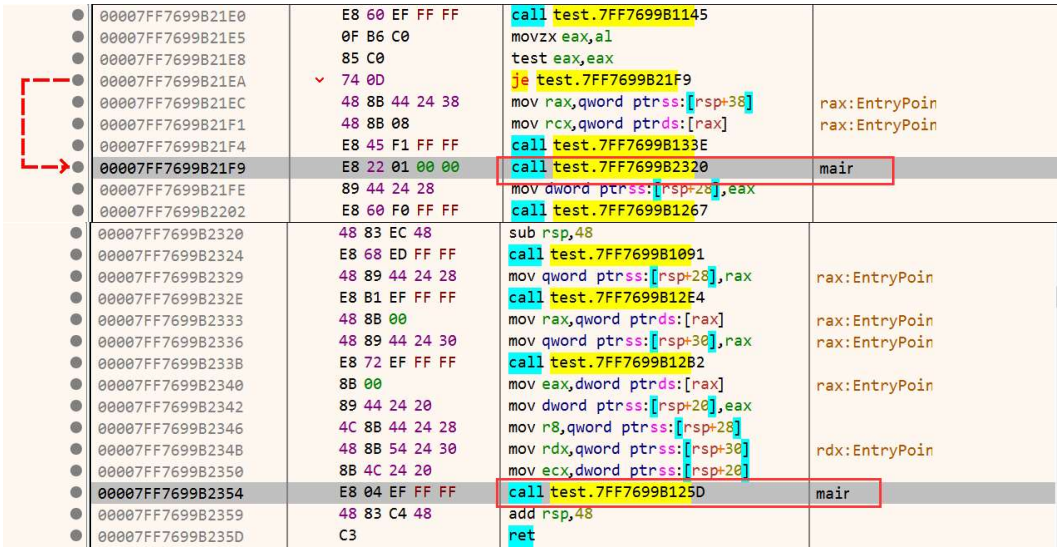
课前会议

64位Release程序的预留空间可能被用来保存寄存器环境（非rcx、rdx、r8、r9寄存器）或者局部变量。

流程控制语句的识别

> **图形识别法**：通过IDA和x64dbg的跳转线条判断流程控制语句，虚线表示条件跳转（jxx），实线表示无条件跳转（jmp），中间包含的每个蓝色圆点表示一条汇编语句，线的开头表示判断开始。

x64dbug定位main函数：

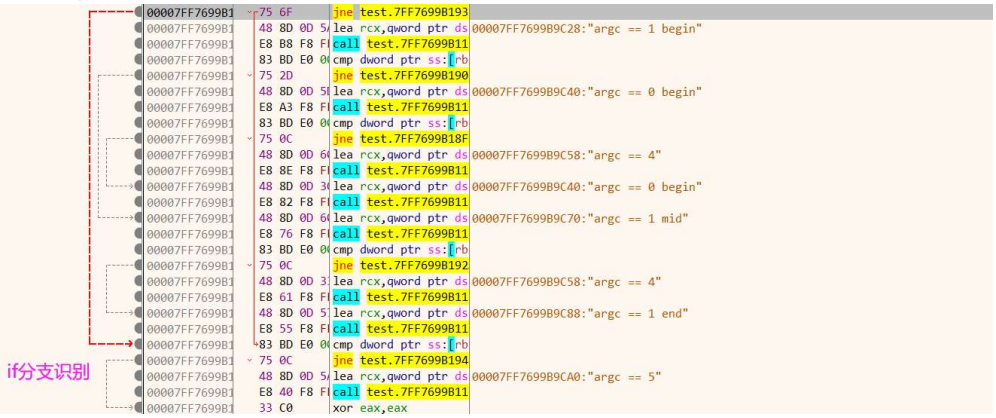


if

识别特征：判断往下跳，上面没有jmp指令

- 单分支，虚线
- 多分支，虚线嵌套

  Debug:



  Release：存在一定的优化，按照结构进行代码还原即可

```
.text:00000001400010A7                jnz     short loc_1400010D4
.text:00000001400010A9                lea     rcx, aArgc1Begin ; "argc == 1 begin"
.text:00000001400010B0                call    sub_140001060
.text:00000001400010B5                lea     rcx, aArgc1Mid  ; "argc == 1 mid"
.text:00000001400010BC                call    sub_140001060
.text:00000001400010C1                lea     rcx, aArgc1End  ; "argc == 1 end"
.text:00000001400010C8                call    sub_140001060
.text:00000001400010CD                xor     eax, eax
.text:00000001400010CF                add     rsp, 28h
.text:00000001400010D3                retn
.text:00000001400010D4 ; ---------------------------------------------------------------------------
.text:00000001400010D4
.text:00000001400010D4 loc_1400010D4:                         ; CODE XREF: main+7↑j
.text:00000001400010D4                cmp     ecx, 5
.text:00000001400010D7                jnz     short loc_1400010E5
.text:00000001400010D9                lea     rcx, aArgc5     ; "argc == 5"
.text:00000001400010E0                call    sub_140001060
.text:00000001400010E5
.text:00000001400010E5 loc_1400010E5:                         ; CODE XREF: main+37↑j
.text:00000001400010E5                xor     eax, eax
.text:00000001400010E7                add     rsp, 28h
.text:00000001400010EB                retn
```

- if 条件为 **&&** 时，ida分析Debug版存在 if 嵌套，**还原成 if 嵌套即可**

```
.text:00000001400118C5                jz      short loc_1400118E5
.text:00000001400118C7                cmp     [rbp+0D0h+arg_0], 2
.text:00000001400118CE                jz      short loc_1400118E5
.text:00000001400118D0                cmp     [rbp+0D0h+arg_0], 3
.text:00000001400118D7                jz      short loc_1400118E5
.text:00000001400118D9                lea     rcx, Format     ; "argc != 1 && argc != 2 && argc != 3"
.text:00000001400118E0                call    printf
.text:00000001400118E5
.text:00000001400118E5 loc_1400118E5:                         ; CODE XREF: sub_140011880+45↑j
.text:00000001400118E5                                        ; sub_140011880+4E↑j ...
.text:00000001400118E5                xor     eax, eax
```

- if 条件为 **||** 时，ida分析Debug版存在两个虚线交叉，**还原成 ||，获取 if 嵌套**

```
.text:00000001400118C5                jz      short loc_1400118D0
.text:00000001400118C7                cmp     [rbp+0D0h+arg_0], 2
.text:00000001400118CE                jnz     short loc_1400118DC
.text:00000001400118D0
.text:00000001400118D0 loc_1400118D0:                         ; CODE XREF: sub_140011880+45↑j
.text:00000001400118D0                lea     rcx, Format     ; "argc == 1 && argc == 2"
.text:00000001400118D7                call    printf
.text:00000001400118DC
.text:00000001400118DC loc_1400118DC:                         ; CODE XREF: sub_140011880+4E↑j
.text:00000001400118DC                xor     eax, eax
```

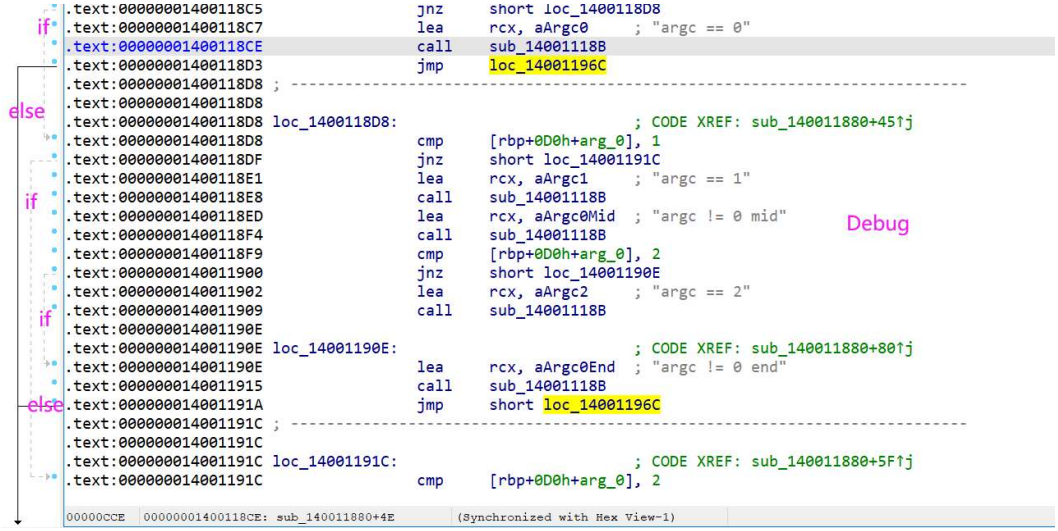
还原为 if 即可

if else

- 虚线 if + 实线 else ，两部分有重叠（重叠部分没有代码，有代码就是goto语句）。

if else-if

- else-if，IDA会将其识别为 else 中嵌套 if。还原代码时，if else-if 结构可按 if else 结构进行还原，也可根据需求将其还原成 if else-if 结构。



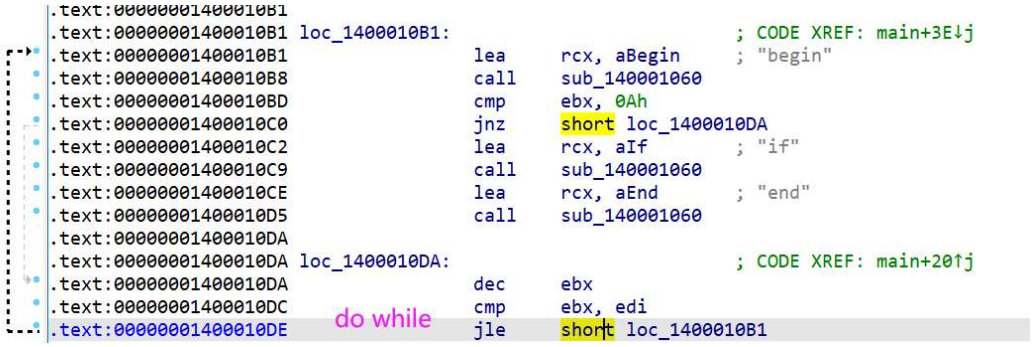
switch-case

> 不适合通过 IDA和x64dbg 查看线结构，case越多，线也就越多。适合通过特征查表进行辨别。

64位中case表中记录的是偏移（4字节）而不是地址。根据case值之间的差值大小会有一个表、两个表、if判断三种方式。

do while

执行完判断后有向上跳转操作（继续执行相关语句，再判断条件）

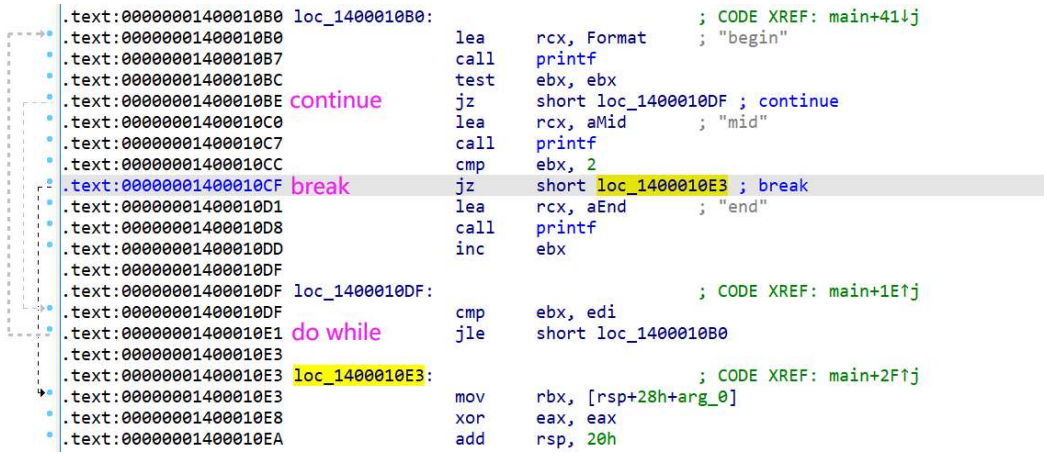


- break：其对应的线条会跳出循环判断条件线条
- continue：线条跳转到语句判断位置

Debug：


```
.text:00000001400118CC                lea     rcx, Format     ; "begin"
.text:00000001400118D3                call    printf
.text:00000001400118D8                cmp     [rbp+110h+var_EC], 0
.text:00000001400118DC                jnz     short loc_1400118E0
.text:00000001400118DE                jmp     short loc_140011908 ; continue
.text:00000001400118E0 ; ---------------------------------------------------------------------------
.text:00000001400118E0
.text:00000001400118E0 loc_1400118E0:                         ; CODE XREF: sub_140011880+5C↑j
.text:00000001400118E0                lea     rcx, aMid       ; "mid"
.text:00000001400118E7                call    printf
.text:00000001400118EC                cmp     [rbp+110h+var_EC], 2
.text:00000001400118F0                jnz     short loc_1400118F4
.text:00000001400118F2                jmp     short loc_140011913 ; break
.text:00000001400118F4 ; ---------------------------------------------------------------------------
.text:00000001400118F4
.text:00000001400118F4 loc_1400118F4:                         ; CODE XREF: sub_140011880+70↑j
.text:00000001400118F4                lea     rcx, aEnd       ; "end"
.text:00000001400118FB                call    printf
.text:0000000140011900                mov     eax, [rbp+110h+var_EC]
.text:0000000140011903                inc     eax
.text:0000000140011905                mov     [rbp+110h+var_EC], eax
.text:0000000140011908
.text:0000000140011908 loc_140011908:                         ; CODE XREF: sub_140011880+5E↑j
.text:0000000140011908                mov     eax, [rbp+110h+arg_0]
.text:000000014001190E                cmp     [rbp+110h+var_EC], eax
.text:0000000140011911                jle     short loc_1400118CC
.text:0000000140011913
.text:0000000140011913 loc_140011913:                         ; CODE XREF: sub_140011880+72↑j
00000CF2  00000001400118F2: sub_140011880+72        (Synchronized with Hex View-1)
```

Release：




```
.text:00000001400010B0 loc_1400010B0:                         ; CODE XREF: main+41↓j
.text:00000001400010B0                lea     rcx, Format     ; "begin"
.text:00000001400010B7                call    printf
.text:00000001400010BC                test    ebx, ebx
.text:00000001400010BE                jz      short loc_1400010DF ; continue
.text:00000001400010C0                lea     rcx, aMid       ; "mid"
.text:00000001400010C7                call    printf
.text:00000001400010CC                cmp     ebx, 2
.text:00000001400010CF                jz      short loc_1400010E3 ; break
.text:00000001400010D1                lea     rcx, aEnd       ; "end"
.text:00000001400010D8                call    printf
.text:00000001400010DD                inc     ebx
.text:00000001400010DF
.text:00000001400010DF loc_1400010DF:                         ; CODE XREF: main+1E↑j
.text:00000001400010DF                cmp     ebx, edi
.text:00000001400010E1                jle     short loc_1400010B0
.text:00000001400010E3
.text:00000001400010E3 loc_1400010E3:                         ; CODE XREF: main+2F↑j
.text:00000001400010E3                mov     rbx, [rsp+28h+arg_0]
.text:00000001400010E8                xor     eax, eax
.text:00000001400010EA                add     rsp, 20h
```

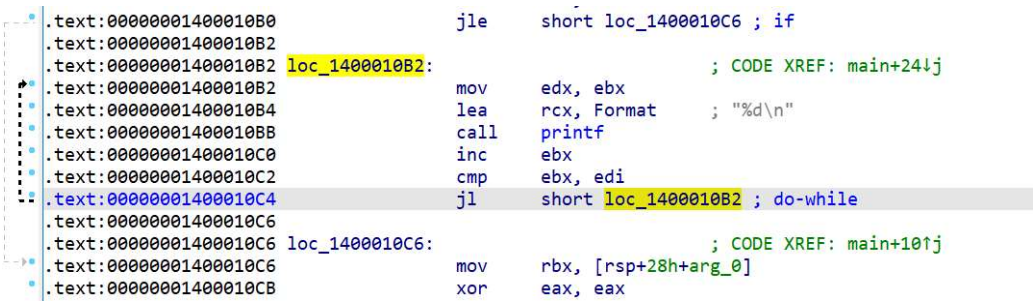
while

Release版编译器会将其优化成 if + do while。

```
.text:00000001400010B0                js      short loc_1400010C6 ; if
.text:00000001400010B2
.text:00000001400010B2 loc_1400010B2:                         ; CODE XREF: main+24↓j
.text:00000001400010B2                mov     edx, ebx
.text:00000001400010B4                lea     rcx, Format     ; "%d\n"
.text:00000001400010BB                call    printf
.text:00000001400010C0                inc     ebx
.text:00000001400010C2                cmp     ebx, edi
.text:00000001400010C4                jle     short loc_1400010B2 ; do-while
.text:00000001400010C6
.text:00000001400010C6 loc_1400010C6:                         ; CODE XREF: main+10↑j
.text:00000001400010C6                mov     rbx, [rsp+28h+arg_0]
.text:00000001400010CB                xor     eax, eax
```

for

Release版编译器会将其优化成 if + do while。


.text:00000001400010B0 jle short loc_1400010C6 ; if
.text:00000001400010B2
.text:00000001400010B2 loc_1400010B2: ; CODE XREF: main+24↓j
.text:00000001400010B2 mov edx, ebx
.text:00000001400010B4 lea rcx, Format ; "%d\n"
.text:00000001400010BB call printf
.text:00000001400010C0 inc ebx
.text:00000001400010C2 cmp ebx, edi
.text:00000001400010C4 jl short loc_1400010B2 ; do-while
.text:00000001400010C6
.text:00000001400010C6 loc_1400010C6: ; CODE XREF: main+10↑j
.text:00000001400010C6 mov rbx, [rsp+28h+arg_0]
.text:00000001400010CB xor eax, eax