

## 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](#)
- [流程控制语句的识别](#)
- [if](#)
- [if else](#)
- [if else-if](#)
- [switch-case](#)
- [do while](#)
- [while](#)
- [for](#)

## 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。

x64dbg定位main函数：

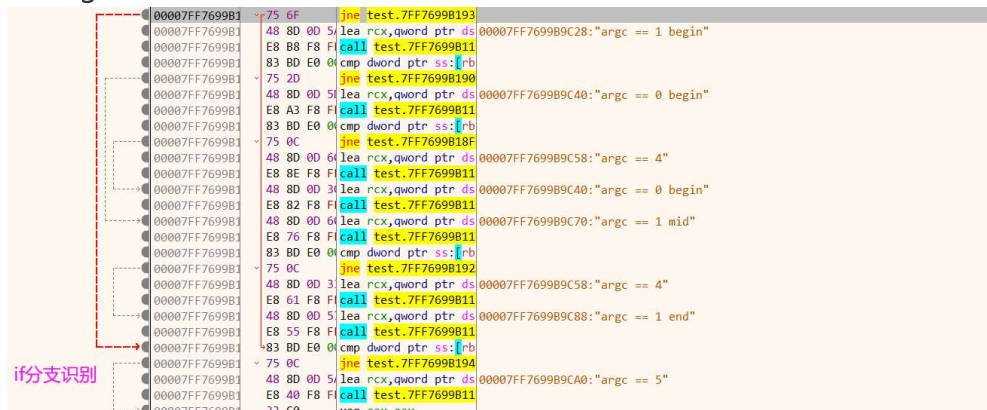
00007FF7699B21E0	E8 60 EF FF FF	call test.7FF7699B1145	
00007FF7699B21E5	0F B6 C0	movzx eax,al	
00007FF7699B21E8	85 C0	test eax,eax	
00007FF7699B21EA	74 0D	je test.7FF7699B21F9	
00007FF7699B21EC	48 8B 44 24 38	mov rax,qword ptrss:[rsp+38]	rax:EntryPoin
00007FF7699B21F1	48 8B 08	mov rcx,qword ptrds:[rax]	rax:EntryPoin
00007FF7699B21F4	E8 45 F1 FF FF	call test.7FF7699B133E	
00007FF7699B21F9	E8 22 01 00 00	call test.7FF7699B2320	mair
00007FF7699B21FE	89 44 24 28	mov dword ptrss:[rsp+28],eax	
00007FF7699B2202	E8 60 F0 FF FF	call test.7FF7699B1267	
00007FF7699B2320	48 83 EC 48	sub rsp,48	
00007FF7699B2324	E8 68 ED FF FF	call test.7FF7699B1091	
00007FF7699B2329	48 89 44 24 28	mov qword ptrss:[rsp+28],rax	rax:EntryPoin
00007FF7699B232E	E8 B1 EF FF FF	call test.7FF7699B12E4	
00007FF7699B2333	48 8B 00	mov rax,qword ptrds:[rax]	rax:EntryPoin
00007FF7699B2336	48 89 44 24 30	mov qword ptrss:[rsp+30],rax	rax:EntryPoin
00007FF7699B233B	E8 72 EF FF FF	call test.7FF7699B12B2	
00007FF7699B2340	8B 00	mov eax,dword ptrds:[rax]	rax:EntryPoin
00007FF7699B2342	89 44 24 20	mov dword ptrss:[rsp+20],eax	
00007FF7699B2346	4C 8B 44 24 28	mov r8,qword ptrss:[rsp+28]	
00007FF7699B234B	48 8B 54 24 30	mov rdx,qword ptrss:[rsp+30]	rdx:EntryPoin
00007FF7699B2350	8B 4C 24 20	mov ecx,dword ptrss:[rsp+20]	
00007FF7699B2354	E8 04 EF FF FF	call test.7FF7699B125D	mair
00007FF7699B2359	48 83 C4 48	add rsp,48	
00007FF7699B235D	C3	ret	

# if

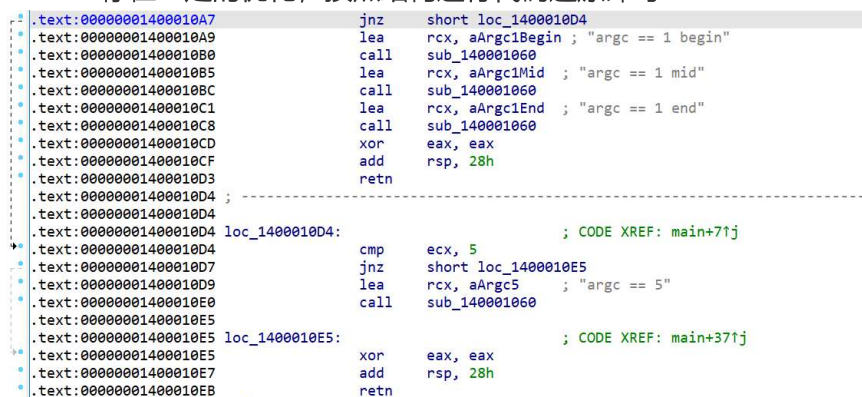
识别特征：判断往下跳，上面没有jmp指令

- 单分支，虚线
- 多分支，虚线嵌套

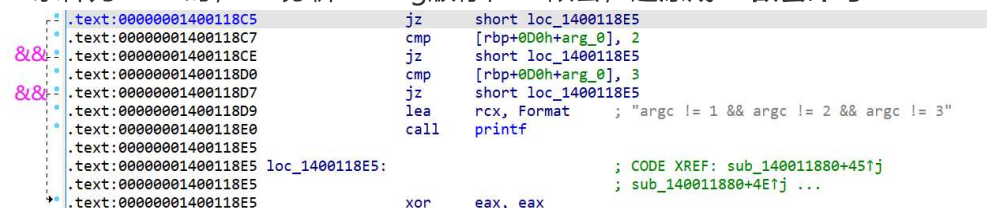
Debug:



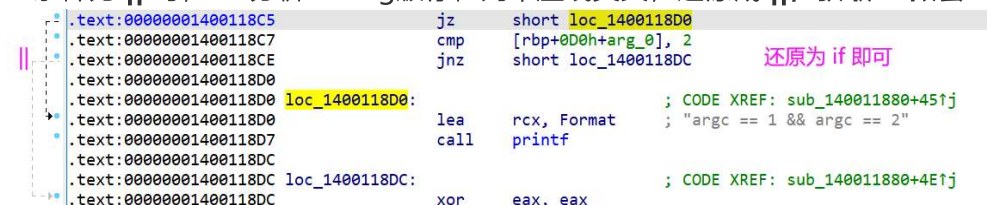
Release: 存在一定的优化，按照结构进行代码还原即可



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



- 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



# 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。

```
.text:000000001400118C5      jnz     short loc_1400118D8
.text:000000001400118C7      lea     rcx, aArgc0      ; "argc == 0"
.text:000000001400118CE      call    sub_14001118B
.text:000000001400118D3      jmp     loc_14001196C
; -----
.text:000000001400118D8      loc_1400118D8:          ; CODE XREF: sub_140011880+45↑j
.text:000000001400118D8      cmp     [rbp+0D0h+arg_0], 1
.text:000000001400118DF      jnz     short loc_14001191C
.text:000000001400118E1      lea     rcx, aArgc1      ; "argc == 1"
.text:000000001400118E8      call    sub_14001118B
.text:000000001400118ED      lea     rcx, aArgc0Mid    ; "argc != 0 mid"
.text:000000001400118F4      call    sub_14001118B
.text:000000001400118F9      cmp     [rbp+0D0h+arg_0], 2
.text:00000000140011900      jnz     short loc_14001190E
.text:00000000140011902      lea     rcx, aArgc2      ; "argc == 2"
.text:00000000140011909      call    sub_14001118B
.text:0000000014001190E      loc_14001190E:          ; CODE XREF: sub_140011880+80↑j
.text:0000000014001190E      lea     rcx, aArgc0End    ; "argc != 0 end"
.text:00000000140011915      call    sub_14001118B
.text:0000000014001191A      jmp     short loc_14001196C
; -----
.text:0000000014001191C      loc_14001191C:          ; CODE XREF: sub_140011880+5F↑j
.text:0000000014001191C      cmp     [rbp+0D0h+arg_0], 2
```

# 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)

```
.text:000000001400010B1      loc_1400010B1:          ; CODE XREF: main+3E↑j
.text:000000001400010B1      lea     rcx, aBegin      ; "begin"
.text:000000001400010B8      call    sub_140001060
.text:000000001400010BD      cmp     ebx, 0Ah
.text:000000001400010C0      jnz     short loc_1400010DA
.text:000000001400010C2      lea     rcx, aIf         ; "if"
.text:000000001400010C9      call    sub_140001060
.text:000000001400010CE      lea     rcx, aEnd        ; "end"
.text:000000001400010D5      call    sub_140001060
.text:000000001400010DA      loc_1400010DA:          ; CODE XREF: main+20↑j
.text:000000001400010DA      dec     ebx
.text:000000001400010DC      cmp     ebx, edi
.text:000000001400010DE      jle     short loc_1400010B1
```

- 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:000000001400118C0      lea     rcx, Format      ; "begin"
.text:000000001400118D3      call    printf
.text:000000001400118D8      cmp     [rbp+110h+var_EC], 0
.text:000000001400118DC      jnz     short loc_1400118E0
.text:000000001400118DE      jmp     short loc_140011908 ; continue
;-----
.text:000000001400118E0      loc_1400118E0:          ; CODE XREF: sub_140011880+5C↑j
.text:000000001400118E0      lea     rcx, aMid        ; "mid"
.text:000000001400118E0      call    printf
.text:000000001400118E7      cmp     [rbp+110h+var_EC], 2
.text:000000001400118EC      jnz     short loc_1400118F4
.text:000000001400118F2      jmp     short loc_140011913 ; break
;-----
.text:000000001400118F4      loc_1400118F4:          ; CODE XREF: sub_140011880+70↑j
.text:000000001400118F4      lea     rcx, aEnd        ; "end"
.text:000000001400118F4      call    printf
.text:00000000140011900      mov     eax, [rbp+110h+var_EC]
.text:00000000140011903      inc     eax
.text:00000000140011905      mov     [rbp+110h+var_EC], eax
.text:00000000140011908      loc_140011908:          ; CODE XREF: sub_140011880+5E↑j
.text:00000000140011908      mov     eax, [rbp+110h+arg_0]
.text:0000000014001190E      cmp     [rbp+110h+var_EC], eax
.text:00000000140011911      jle     short loc_1400118C0
.text:00000000140011913      loc_140011913:          ; CODE XREF: sub_140011880+72↑j
;-----
000000C02 000000001400118F2: sub_140011880+72      (Synchronized with Hex View-1)
```

Release:

```
.text:00000000140010B0      loc_140010B0:          ; CODE XREF: main+41↑j
.text:00000000140010B0      lea     rcx, Format      ; "begin"
.text:00000000140010B7      call    printf
.text:00000000140010B8      test    ebx, ebx
.text:00000000140010BE      continue      jz     short loc_140010DF ; continue
.text:00000000140010C0      lea     rcx, aMid        ; "mid"
.text:00000000140010C7      call    printf
.text:00000000140010CC      cmp     ebx, 2
.text:00000000140010CF      break      jz     short loc_140010E3 ; break
.text:00000000140010D1      lea     rcx, aEnd        ; "end"
.text:00000000140010D8      call    printf
.text:00000000140010DD      inc     ebx
.text:00000000140010DF      loc_140010DF:          ; CODE XREF: main+1E↑j
.text:00000000140010DF      cmp     ebx, edi
.text:00000000140010E1      do while      jle     short loc_140010B0
.text:00000000140010E3      loc_140010E3:          ; CODE XREF: main+2F↑j
.text:00000000140010E3      mov     rbx, [rsp+28h+arg_0]
.text:00000000140010E8      xor     eax, eax
.text:00000000140010EA      add     rsp, 20h
```

## while

Release版编译器会将其优化成 if + do while。

```
.text:00000000140010B0      js      short loc_140010C6 ; if
.text:00000000140010B2      loc_140010B2:          ; CODE XREF: main+24↑j
.text:00000000140010B2      mov     edx, ebx
.text:00000000140010B4      lea     rcx, Format      ; "%d\n"
.text:00000000140010B8      call    printf
.text:00000000140010C0      inc     ebx
.text:00000000140010C2      cmp     ebx, edi
.text:00000000140010C4      jle     short loc_140010B2 ; do-while
.text:00000000140010C6      loc_140010C6:          ; CODE XREF: main+10↑j
.text:00000000140010C6      mov     rbx, [rsp+28h+arg_0]
.text:00000000140010CB      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:00000001400010B8      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

```