2021/05/28_x64汇编与逆向_第5课_异常处理的识别

笔记本： x64汇编与逆向
创建时间： 2021/5/28 星期五 10:00
作者： ileemi

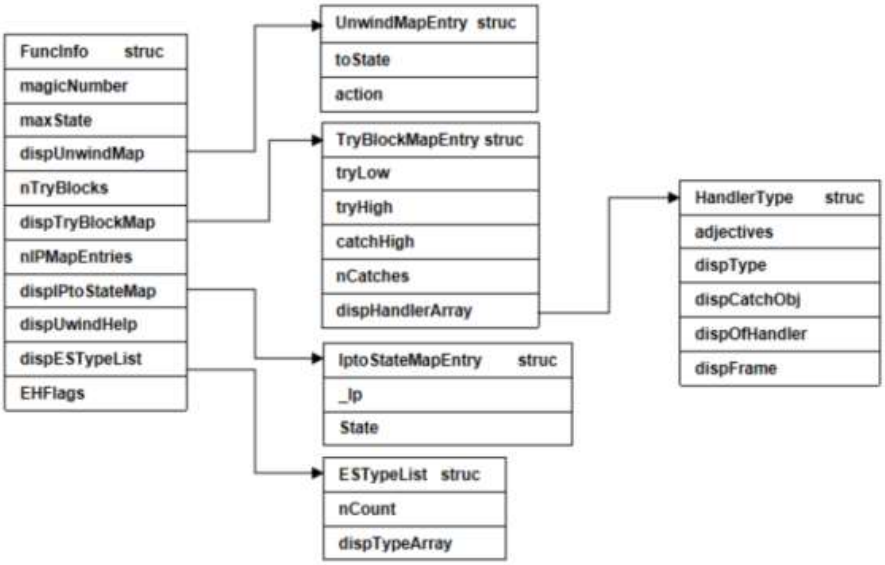
课前会议

平坦优化：代码混淆中经常见到的行为。通过空间换时间（重复相关操作）。

代码预执行（InterCPU硬件支持）

异常处理

> 64位异常相关结构体表：



32位程序使用SEH注册异常，通过SEH注册异常，其效率较低（函数入口注册，函数出口注销），当函数被多次调用时就需要多次执行注册、注销操作。

解决上述效率问题，微软做法就是作一个全局的异常表，存储各个函数对应的异常处理（由系统注册，存储在PE文件中。多做一个节（.pdata），用来存储整个软件所有函数的异常处理相关结构体）。

> VS使用的异常处理表版本：

- VS2013之前使用 SEH
- VS2013 x64 __CxxFrameHandler3 FH3
- VS2019 update1 x64 __CxxFrameHandler3 FH3
- VS2019 update2 x64 __CxxFrameHandler4 FH3 FH4
- VS2019 update3 x64 __CxxFrameHandler4 FH4

异常相关结构体

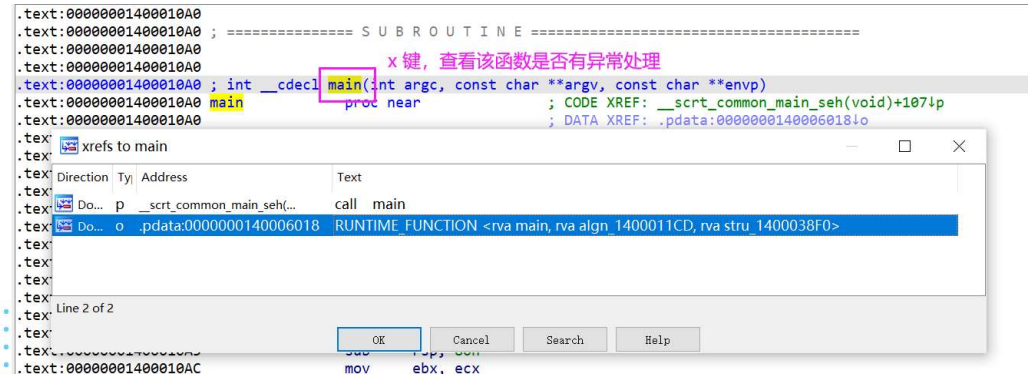
RUNTIME_FUNCTION

```
RUNTIME_FUNCTION struc ; (sizeof=0xC, mappedto_3)
FunctionStart    dd ?    ; offset rva
FunctionEnd      dd ?    ; offset rva pastend
UnwindInfo       dd ?    ; offset rva
RUNTIME_FUNCTION ends
```

- FunctionStart：函数起始地址
- FunctionEnd：函数结束地址
- UnwindInfo：展开信息地址（指向异常处理结构体（UNWIND_INFO））

表结构成员偏移为RVA的好处：表的大小可以更小，支持随机基址。

> X 键查看该函数是否存在异常处理：

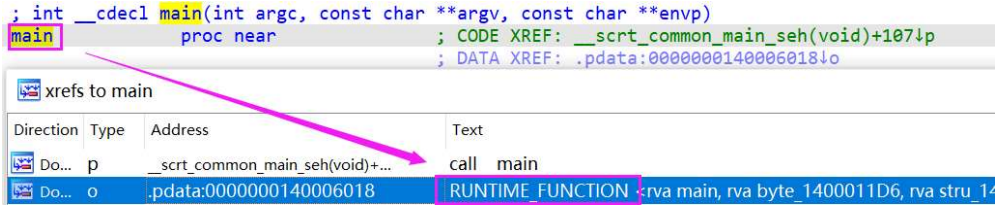


FuncInfo

```
FuncInfo        struc ; (sizeof=0x28, mappedto_6)
magicNumber     dd ?    ; base 16
maxState        dd ?    ; base 10
pUnwindMap      dd ?    ; offset rva
nTryBlocks      dd ?    ; base 10
pTryBlockMap    dd ?    ; offset rva
nIPMapEntries   dd ?    ; base 10
pIPtoStateMap   dd ?    ; offset rva
dispUnwindHelp  dd ?    ; base 10
pESTypeList     dd ?    ; offset rva
EHFlags         dd ?    ; base 16
FuncInfo        ends
```

通过函数引用定位异常信息并还原代码

- 通过该方法可以定位对应 RUNTIME_FUNCTION



- 通过 RUNTIME_FUNCTION 参数3定位 UNWIND_INFO_HDR结构体

```
RUNTIME_FUNCTION <rva main, rva byte_1400011D6, rva stru_1400038F0>
```

- 通过 UNWIND_INFO_HDR 结构体的参数4及参数5确定__CxxFrameHandler3 的版本以及 FuncInfo 结构信息。

```
stru_1400038F0  UNWIND_INFO_HDR <19h, 15h, 2, 0>
                                        ; DATA XREF: .pdata:0000000140006018↓o
                UNWIND_CODE <0Ch, 0F2h> ; UWOP_ALLOC_SMALL
                UNWIND_CODE <5, 30h>    ; UWOP_PUSH_NONVOL
                dd rva __CxxFrameHandler3
                dd rva stru_1400032F0
```

FuncInfo

- 通过 FuncInfo 结构的参数4以及参数5可以确定该函数中对应的 try-catch 数量以及TryBlockMapEntry结构信息。

```
stru_1400032F0  FuncInfo <19930522h, 4, rva stru_140003900, 2, rva stru_140003920, 3, \
                                        ; DATA XREF: .rdata:00000001400038FC↓o
                                        ; .rdata:00000001400039E4↓o
                        rva stru_1400039C0, 72, 0, 1>
```

TryBlockMapEntry

- 通过 TryBlockMapEntry 结构信息可以获取 try-catch 对应的 catch 个数以及 HandlerType 结构信息。
- 通过 HandlerType 结构信息获取对应的 catch 实现代码。

```
stru_140003920  TryBlockMapEntry <0, 0, 1, 3, rva stru_140003948>
                                        ; DATA XREF: .rdata:stru_1400032F0↑o
                TryBlockMapEntry <2, 2, 3, 3, rva stru_140003984>
stru_140003948  HandlerType <0, rva ??_R0H@8, 80, rva loc_140001EF0, 56>
                                        ; DATA XREF: .rdata:stru_140003920↑o
                                        ; int `RTTI Type Descriptor'
                HandlerType <0, rva ??_R0N@8, 88, rva loc_140001F18, 56> ; double `RTTI Type
                HandlerType <0, rva ??_R0J@8, 96, rva loc_140001F40, 56> ; __int64 `RTTI Ty
stru_140003984  HandlerType <0, rva ??_R0H@8, 104, rva loc_140001F68, 56>
                                        ; DATA XREF: .rdata:00000001400039341↑o
                                        ; int `RTTI Type Descriptor'
                HandlerType <0, rva ??_R0N@8, 112, rva loc_140001F90, 56> ; double `RTTI Typ
                HandlerType <0, rva ??_R0J@8, 120, rva loc_140001FB8, 56> ; __int64 `RTTI T
```

指向catch实现代码

- 通过 FuncInfo 结构的参数获取 try 的范围，根据上下文的汇编代码进行还原即可。

```
stru_1400032F0  FuncInfo <19930522h, 4, rva stru_140003900, 2, rva stru_140003920, 3, \
                                        ; DATA XREF: .rdata:00000001400038FC↓o
                                        ; .rdata:00000001400039E4↓o
                        rva stru_1400039C0, 72, 0, 1>
```

IPtoStateMap

高版本中，当两个try-catch连续时，第一个try-catch可能不会显示对应的结束表IPtoStateMap

```
stru_1400039C0  IPtoStateMap <rva loc_14000113E, 0>
                                        ; DATA XREF: .rdata:stru_1400032F0↑o
                IPtoStateMap <rva loc_14000117C, 2>
                IPtoStateMap <rva byte_1400011D5, -1>
```

第二个 try-catch起始地址

第一个 try-catch起始地址

第二个 try-catch结束地址

UNWIND_INFO_HDR

```
UNWIND_INFO_HDR struc ; (sizeof=0x4, mappedto_4)
Ver3_Flags      db ?    ; base 16
PrologSize      db ?    ; base 16
CntUnwindCodes  db ?    ; base 16
FrReg_FrRegOff  db ?    ; base 16
UNWIND_INFO_HDR ends
```

- Ver3_Flags： 低3位Version，高5位Flags
  Flags标识存在 UNW_LFAG_EHANDLER(1)，则 UNWIND_CODE 结构后会跟随一个函数地址和一个 FuncInfo
- PrologSize： 序言大小（在函数入口保存环境代码的字节大小）
- CntUnwindCodes： 展开代码数组大小，标识后续跟着多少个 UNWIND_CODE 结构，必须2字节对齐，不对齐地址用零填充。

```
.rdata:00000001400039CC stru_1400039CC  UNWIND_INFO_HDR <1, 9, 1, 0>
.rdata:00000001400039CC                                 ; DATA XREF: .pdata:0000000140006084↓o
.rdata:00000001400039D0                 UNWIND_CODE <9, 62h>    ; UWOP_ALLOC_SMALL
.rdata:00000001400039D2                 align 4
```

- FrReg_FrRegOff： 低4位为帧寄存器，高4位为帧寄存器偏移量

```
.rdata:00000001400038F0 stru_1400038F0  UNWIND_INFO_HDR <19h, 0Ch, 2, 0>
.rdata:00000001400038F0                                         ; DATA XREF: .pdata:0000000140006018↓o
.rdata:00000001400038F4                 UNWIND_CODE <0Ch, 0F2h> ; UWOP_ALLOC_SMALL
.rdata:00000001400038F6                 UNWIND_CODE <5, 30h>    ; UWOP_PUSH_NONVOL
.rdata:00000001400038F8                 dd rva __CxxFrameHandler4 ; 异常处理函数
.rdata:00000001400038FC                 dd rva byte_140003900   ; 记录FUNCTION地址(RVA)
.rdata:0000000140003900 byte_140003900  db 38h                  ; DATA XREF: .rdata:00000001400038FC↑o
.rdata:0000000140003901                 db  0Dh
.rdata:0000000140003902                 db  39h ; 9
.rdata:0000000140003903                 db    0
.rdata:0000000140003904                 db    0
.rdata:0000000140003905                 db  12h
.rdata:0000000140003906                 db  39h ; 9
.rdata:0000000140003907                 db    0
.rdata:0000000140003908                 db    0
.rdata:0000000140003909                 db  65h ; e
.rdata:000000014000390A                 db  39h ; 9
```

> 在VS中将 __CxxFrameHandler4 修改为 __CxxFrameHandler3 需要在项目属性 --> C/C++ --> 命令行 --> **/d2FH4-**(关闭__CxxFrameHandler4)。注意：/d2FH4 开启 __CxxFrameHandler4。

__CxxFrameHandler3

> 相比32位的 FuncInfo（共5个成员），64位 FuncInfo 结构体（共10个成员，大部分记录展开信息，**pIPtoStateMap**指向新增加的表 **Ipto StateMapEntry**）扩展了一些成员。

```
FuncInfo        struc ; (sizeof=0x28, mappedto_6)
magicNumber     dd ?    ; base 16
maxState        dd ?    ; base 10
pUnwindMap      dd ?    ; offset rva
nTryBlocks      dd ?    ; base 10
pTryBlockMap    dd ?    ; offset rva
nIPMapEntries   dd ?    ; base 10
pIPtoStateMap   dd ?    ; offset rva
dispUnwindHelp  dd ?    ; base 10
pESTypeList     dd ?    ; offset rva
EHFlags         dd ?    ; base 16
FuncInfo        ends
```

```
.rdata:00000001400032F0 stru_1400032F0  FuncInfo <19930522h, 4, rva stru_140003900, 2, rva stru_140003920, 3, \
.rdata:00000001400032F0                                         ; DATA XREF: .rdata:00000001400038FC↓o
.rdata:00000001400032F0                                         ; .rdata:00000001400039E4↓o
.rdata:00000001400032F0                         rva stru_1400039C0, 72, 0, 1>
```

IPtoStateMap

新增加(主要)

IPtoStateMap

结构如下：该结构体可以确定try-catch "try" 块的范围。

```
IPtoStateMap    struc ; (sizeof=0x8, mappedto_11)
pc              dd ?    ; offset rva
state   dd ?    ; base 10
IPtoStateMap    ends
```

- pc：EIP
- start：下标（ida识别try-catch范围就是依靠该下标）

```
.rdata:00000001400039C0 stru_1400039C0  IPtoStateMap <rva loc_14000113E, 0>
.rdata:00000001400039C0                                         ; DATA XREF: .rdata:stru_1400032F0↑o
.rdata:00000001400039C8                 IPtoStateMap <rva loc_14000117C, 2>
.rdata:00000001400039D0                 IPtoStateMap <rva byte_1400011D5, -1>
```

TryBlockMapEntry

```
TryBlockMapEntry struc ; (sizeof=0x14, mappedto_8)
tryLow   dd ?    ; base 10
tryHigh  dd ?    ; base 10
catchHigh        dd ?    ; base 10
nCatches         dd ?    ; base 10
pHandlerArray    dd ?    ; offset rva
TryBlockMapEntry ends
```

- nCatches：case数量
- pHandlerArray：指向 HandlerType 结构体

HandlerType

```
HandlerType      struc ; (sizeof=0x14, mappedto_9)
adjectives       dd ?    ; base 16
pType    dd ?    ; offset rva
dispCatchObj     dd ?    ; base 10
addressOfHandler dd ?    ; offset rva
dispFrame        dd ?    ; base 10
HandlerType      ends
```

- adjectives：类型
- pType：RTTI
- dispCatchObj：catch对象
- addressOfHandler：Handler地址（catch实现代码位置）

```
stru_140003948  HandlerType <0, rva ??_R0H@8, 80, rva loc_140001EF0, 56>
                                        ; DATA XREF: .rdata:stru_1400039201o
                                        ; int `RTTI Type Descriptor'
                HandlerType <0, rva ??_R0N@8, 88, rva loc_140001F18, 56> ; double `RTTI Type Descriptor'
                HandlerType <0, rva ??_R0_J@8, 96, rva loc_140001F40, 56> ; __int64 `RTTI Type Descriptor
.data:0000000140005050 ; int `RTTI Type Descriptor'
.data:0000000140005050 ??_R0H@8        dq offset ??_7type_info@@6B@; pVFTable
.data:0000000140005050                                         ; DATA XREF: .rdata:stru_1400039481o
.data:0000000140005050                                         ; .rdata:stru_140003984↑o ...
.data:0000000140005050                 dq 0                    ; spare ; reference to RTTI's vftable
.data:0000000140005050                 db '.H',0               ; name
.data:0000000140005063                 align 8
.data:0000000140005068 ; __int64 `RTTI Type Descriptor'
.data:0000000140005068 ??_R0_J@8       dq offset ??_7type_info@@6B@; pVFTable
.data:0000000140005068                                         ; DATA XREF: .rdata:00000001400039701o
.data:0000000140005068                                         ; .rdata:00000001400039AC↑o ...
.data:0000000140005068                 dq 0                    ; spare ; reference to RTTI's vftable
.data:0000000140005068                 db '._J',0              ; name
.data:000000014000507C                 align 20h
.data:0000000140005080 ; public class type_info /* mdisp:0 */
.data:0000000140005080 ; class type_info `RTTI Type Descriptor'
.data:0000000140005080 ??_R0?AVtype_info@@@8 dq offset ??_7type_info@@6B@
.data:0000000140005080                                         ; DATA XREF: .rdata:000000014000350C↑o
.data:0000000140005080                                         ; .rdata:type_info::`RTTI Base Class Descriptor at
.data:0000000140005080                                         ; reference to RTTI's vftable
.data:0000000140005088                 dq 0                    ; internal runtime reference
.data:0000000140005090 aAvtypeInfo     db '.?AVtype_info@@',0  ; type descriptor name
```

识别catch处理的类型

通过函数内部Throw定位catch实现代码

1. 通过_ThrowInfo 参数4 rva值 定位catchable具体实现

```
loc_140001123:                          ; CODE XREF: main+37↑j
                mov     rax, 1234567812345678h
                mov     [rsp+88h+pExceptionObject], rax
                lea     rdx, __TI1_J    ; pThrowInfo
                lea     rcx, [rsp+88h+pExceptionObject] ; pExceptionObject

loc_14000113E:                          ; DATA XREF: .rdata:stru_1400039C0↓o
;   try {
                call    _CxxThrowException
```

双击定位RTTI

```
.rdata:0000000140003AC0 __TI1_J         _ThrowInfo <0, 0, 0, 3B00h>
.rdata:0000000140003AC0                                        ; DATA XREF: main+92↑o
.rdata:0000000140003AC0                                        ; main+EB↑o
.rdata:0000000140003AC0                                        ; attributes
.rdata:0000000140003AD0                 align 20h
.rdata:0000000140003AE0 ; const _ThrowInfo _TI1H
.rdata:0000000140003AE0 __TI1H          _ThrowInfo <0, 0, 0, 3BB8h>
.rdata:0000000140003AE0                                        ; DATA XREF: main+CA↑o
.rdata:0000000140003AE0                                        ; main+123↑o
.rdata:0000000140003AE0                                        ; attributes
.rdata:0000000140003AF0                 align 20h
.rdata:0000000140003B00 __CTA1_J        dd 1                   ; count of catchable type addresses following
.rdata:0000000140003B04                 dd rva __CT??_R0_J@8   ; catchable type '__int64'
.rdata:0000000140003B08                 align 10h
```

双击

2. 定位 RTTI

```
.rdata:0000000140003B70 __CT??_R0_J@8   dd CT_IsSimpleType      ; DATA XREF: .rdata:00000001400003B04↑o
.rdata:0000000140003B70                                         ; attributes
.rdata:0000000140003B74                 dd rva ??_R0_J@8        ; __int64 `RTTI Type Descriptor'
.rdata:0000000140003B78                 dd 0                    ; mdisp
.rdata:0000000140003B7C                 dd -1                   ; pdisp
.rdata:0000000140003B80                 dd 0                    ; vdisp
.rdata:0000000140003B84                 dd 8                    ; size of thrown object
.rdata:0000000140003B88                 dd 0                    ; reference to optional copy constructor
.rdata:0000000140003B8C                 db    0
.rdata:0000000140003B8D                 db    0
.rdata:0000000140003B8E                 db    0
.rdata:0000000140003B8F                 db    0
```

双击定位 RTTI

3. 查找引用定位对应的 HandlerType 表

```
.data:0000000140005068 ??_R0_J@8        dq offset ??_7type_info@@6B@; pVFTable
.data:0000000140005068                                         ; DATA XREF: .rdata:00000001400039701o
.data:0000000140005068                                         ; .rdata:00000001400039AC↑o ...
.data:0000000140005068                 dq 0                    ; spare ; reference to RTTI's vftable
.data:0000000140005068                 db '._J',0              ; name
.data:000000014000507C                 align 20h
.data:0000000140005080 ; public class type_info /* mdisp:0 */
.data:0000000140005080 ; class type_info `RTTI Type Descriptor'
```

查找引用定位对应的 HandlerType 表

xrefs to __int64 `RTTI Type Descriptor'

Direction	Ty	Address	Text
Up	o	.rdata:0000000140003970	HandlerType <0, rva ??_R0_J@8, 96, rva loc_140001F40, 56>; __int64 `RTTI Type Descriptor'
Up	o	.rdata:00000001400039AC	HandlerType <0, rva ??_R0_J@8, 120, rva loc_140001FB8, 56>; __int64 `RTTI Type Descriptor'
Up	o	.rdata:0000000140003B74	dd rva ??_R0_J@8; __int64 `RTTI Type Descriptor'

4. 通过 HandlerType 表参数4定位catch具体的实现代码

```
HandlerType <0, rva ??_R0_J@8, 96, rva loc_140001F40, 56> ; __int64 `RTTI
```

```
.text:0000000140001F40 ; __unwind { // __CxxFrameHandler3
.text:0000000140001F40 ;   catch(__int64) // owned by 14000113E
.text:0000000140001F40                 mov     [rsp+88h+var_78], rdx
.text:0000000140001F45                 push    rbp
.text:0000000140001F46                 sub     rsp, 20h
.text:0000000140001F4A                 mov     rbp, rdx
.text:0000000140001F4D                 lea     rcx, aLongLong  ; "long long \n"
.text:0000000140001F54                 call    sub_140001060
.text:0000000140001F59                 nop
.text:0000000140001F5A                 lea     rax, loc_1400010DB
.text:0000000140001F61                 add     rsp, 20h
.text:0000000140001F65                 pop     rbp
.text:0000000140001F66                 retn
```

> IPtoStateMap 参数2的值可用来匹配 TryBlockMapEntry 参数1的值，用来确定对应的 catch 代码：

```
stru_1400039C0  IPtoStateMap <rva loc_14000113E, 0>
                                        ; DATA XREF: .rdata:stru_1400032F0↑o
                IPtoStateMap <rva loc_14000117C, 2>
                IPtoStateMap <rva byte_1400011D5, -1>
stru_140003920  TryBlockMapEntry <0, 0, 1, 3, rva stru_140003948>
                                        ; DATA XREF: .rdata:stru_1400032F0↑o
                TryBlockMapEntry <2, 2, 3, 3, rva stru_140003984>
stru_140003948  HandlerType <0, rva ??_R0H@8, 80, rva loc_140001EF0, 56>
                                        ; DATA XREF: .rdata:stru_140003920↑o
                                        ; int `RTTI Type Descriptor'
                HandlerType <0, rva ??_R0N@8, 88, rva loc_140001F18, 56> ; double `RTTI Type De
                HandlerType <0, rva ??_R0_J@8, 96, rva loc_140001F40, 56> ; __int64 `RTTI Type
```

__CxxFrameHandler4

> 相关结构体声明在 ehdata4.h 和 ehdata4_export.h 头文件中。

解决__CxxFrameHandler3中 .pdata节 ExceptionDir 中存储 RUNTIME_FUNCTION 信息较多的情况（占用空间较大），在 __CxxFrameHandler4 中将原有的所有结构体都该为了变长结构体，结构体中的有效成员数量不确定。所有成员都有下面的位段结构：用来表示结构体中有效的成员

```
struct FuncInfoHeader
{
    union
    {
#pragma warning(push)
#pragma warning(disable: 4201) // nonstandard extension used: nameless struct/union
        struct
        {
            uint8_t isCatch        : 1;  // 1 if this represents a catch funclet, 0 otherwise
            uint8_t isSeparated    : 1;  // 1 if this function has separated code segments, 0 otherwise
            uint8_t BBT            : 1;  // Flags set by Basic Block Transformations
            uint8_t UnwindMap      : 1;  // Existence of Unwind Map RVA
            uint8_t TryBlockMap    : 1;  // Existence of Try Block Map RVA
            uint8_t EHs            : 1;  // EHs flag set
            uint8_t NoExcept       : 1;  // NoExcept flag set
            uint8_t reserved       : 1;
        };
#pragma warning(pop)
        uint8_t value;
    };
```

FuncInfo4

```
.rdata:00000001400038FC                 dd rva unk_140003900    ; FuncInfo4
.rdata:0000000140003900 unk_140003900   db  38h ; 8            ; DATA XREF: .rdata:00000001400038FC↑o
.rdata:0000000140003900                                         ; FuncInfo4
.rdata:0000000140003901                 dd 390Dh                ; dispUnwindMap
.rdata:0000000140003905                 dd 3912h                ; dispTryBlockMap
.rdata:0000000140003909                 dd 3965h                ; dispIPtoStateMap
.rdata:000000014000390D                 db    8
.rdata:000000014000390E                 db    8
.rdata:000000014000390F                 db  10h
.rdata:0000000140003910                 db  18h
static_assert(sizeof(FuncInfoHeader) == sizeof(uint8_t), "Size of FuncInfoHeader not 1 Byte");

struct FuncInfo4
{
    FuncInfoHeader      header;
    uint32_t            bbtFlags;           // flags that may be set by BBT processing

    int32_t             dispUnwindMap;      // Image relative offset of the unwind map
    int32_t             dispTryBlockMap;    // Image relative offset of the handler map
    int32_t             dispIPtoStateMap;   // Image relative offset of the IP to state map
    uint32_t            dispFrame;          // displacement of address of function frame wrt establisher frame

    FuncInfo4()
    {
        header.value = 0;
        bbtFlags = 0;
        dispUnwindMap = 0;
        dispTryBlockMap = 0;
        dispIPtoStateMap = 0;
        dispFrame = 0;
    }

};
```

FuncInfo4

38H: 0011 1000

HandlerTypeHeader

```
struct HandlerTypeHeader
{
  // See contAddr for description of these values
  enum contType
  {
    NONE = 0b00,
    ONE = 0b01,
    TWO = 0b10,
    RESERVED = 0b11
  };
  union
  {
#pragma warning(push)
#pragma warning(disable: 4201) // nonstandard extension used: nameless struct/union
  struct
  {
    uint8_t adjectives   : 1; // Existence of Handler Type adjectives (bitfield)
    uint8_t dispType     : 1; // Existence of Image relative offset of the corresponding type descriptor
    uint8_t dispCatchObj : 1; // Existence of Displacement of catch object from base
    uint8_t contIsRVA    : 1; // Continuation addresses are RVAs rather than function relative, used for separated code
    uint8_t contAddr     : 2; // 1.   00: no continuation address in
```

```
    metadata, use what the catch funclet returns
        // 2.    01: one function-relative continuation address
        // 3.    10: two function-relative continuation addresses
        // 4.    11: reserved
        uint8_t unused       : 2;
    };
#pragma warning(pop)
    uint8_t value;
    };
};
```

```
.rdata:0000000140003921                 db    6                 ; HandlerMap1
.rdata:0000000140003922                 db  16h                                 16H: 0001 0110
.rdata:0000000140003923                 dd 5050h                ; catch1
.rdata:0000000140003927                 db  90h
.rdata:0000000140003928                 dd 1EE0h                ; catch1对应的处理代码 -- "catch int\n"
.rdata:000000014000392C                 db  64h ; d
.rdata:000000014000392D                 db  16h
.rdata:000000014000392E                 dd 5038h                ; catch2
.rdata:0000000140003932                 db 0A0h
.rdata:0000000140003933                 dd 1F0Bh                ; catch2对应的处理代码 -- "catch double\n"
.rdata:0000000140003937                 db 0E8h
.rdata:0000000140003938                 db  16h
.rdata:0000000140003939                 dd 5068h                ; catch3
.rdata:000000014000393D                 db 0B0h
.rdata:000000014000393E                 dd 1F36h                ; catch3对应的处理代码 -- "catch long long\n"
.rdata:0000000140003942                 db  64h ; d
.rdata:0000000140003943                 db    6                 ; HandlerMap2
.rdata:0000000140003944                 db  16h
.rdata:0000000140003945                 dd 5050h                ; catch1
.rdata:0000000140003949                 db 0C0h
.rdata:000000014000394A                 dd 1F61h                ; catch1对应的处理代码 -- "catch int\n"
.rdata:000000014000394E                 db 0ECh
.rdata:000000014000394F                 db  16h
.rdata:0000000140003950                 dd 5038h                ; catch2
.rdata:0000000140003954                 db 0D0h
.rdata:0000000140003955                 dd 1F8Ch                ; catch2对应的处理代码 -- "catch double\n"
.rdata:0000000140003959                 db 0F0h
```

IPtoStateMap

记录函数中异常开始的偏移值，不在记录rva，函数地址有编译器记录。

> 通过下面方式可计算try-catch起始地址以及结束地址：

```
.rdata:00000001400028BA                 db    4                 ; IPtoStateMap
.rdata:00000001400028BB                 db 0AEh                 ; AE >> 1 = 57 + 1400010A0(main函数首地址) = 1400010F7
.rdata:00000001400028BC                 db    2
.rdata:00000001400028BD                 db  7Ch ; |              ; 7C >> 1 = 3E + 1400010F7 = 140001135
.rdata:00000001400028BE                 db    0
.rdata:00000001400028BF                 db    0
```