结构体

当

1. 结构体变量传参
2. 结构体变量的指针传参
3. 结构体变量返回
4. 结构体变量的指针返回
5. 有明显的对齐行为

满足其一则可以识别出是结构体，不然跟普通的变量没区别

识别

- 成员`.`访问
  - 存在常量折叠
- 指针`->`访问
  - 直接从`首地址+偏移`访问
- 传参
  - 结构体变量的指针
    - 首地址+偏移
  - 结构体变量
    - 结构体比较小，挨个push

```
struct Test {
    int a;
    int b;
};

.text:004011D2                 push    [ebp+var_30]
.text:004011D5                 push    [ebp+var_34] // 结构体开头
.text:004011D8                 call    show_test1
```

    - 结构体不太大，将push操作转化为等价向栈上的`memcpy`，标志`mov edi, esp`，高版本利用`xmm0`向栈上传递

```
struct Person {
    char name[13];      // +0
    short int age;      // 14
    char gender;        // +16
    double wight;       // +24
    float height;       // +32
    int num;            // +36  size: 40
};

// 在高版本VS上
```

```
movups  xmm0, [ebp+var_2C]
mov     eax, esp                        // 以下往栈中拷贝
movups  xmmword ptr [eax], xmm0
movups  xmm0, [ebp+var_1C]
movups  xmmword ptr [eax+10h], xmm0
movq    xmm0, [ebp+var_C]
movq    qword ptr [eax+20h], xmm0
call    show_person1
```

- 结构体比较大，高版本也使用向栈上的`memcpy`，标志`mov edi, esp`

```
sub     esp, 6Ch            ; 向上抬栈
lea     esi, [ebp+var_CC]    ; 结构体开头
mov     ecx, 26h            ; 拷贝次数
mov     edi, esp            ; 目标：栈上，此为重要判定标志
rep movsd
```

- 返回值
  - int
    - eax
  - __int64
    - edx.eax
  - float
    - 低版本 st0 <- dword ptr
    - 高版本可能 xmm0 <- dword ptr
  - double
    - 低版本 st0 <- qword ptr
    - 高版本可能 xmm0 <- qword ptr
  - 结构体对象
    - 比较小，通过edx.eax往栈上的临时变量空间拷贝，即返回临时对象，在进行变量之间的拷贝

```
; 在函数内部
; ...
mov     eax, [ebp+var_8]
add     esp, 0Ch
mov     edx, [ebp+var_4]
; ...

; 返回后
call    ret_test
push    edx
push    eax
push    offset aDD_0    ; "%d %d\n"
call    printf
```

    - 比较大，额外隐藏加了一个参数，此参数就是返回值，eax存放这个对象的指针，依然会产生临时对象

```
lea     eax, [ebp-68h]  // 隐含参数，结构体指针，在栈上有此参数的结构体(临时对象)的空间
push    eax
call    ret_person
movups  xmm0, xmmword ptr [eax]          // 返回后eax为临时对象的指针
movups  xmmword ptr [ebp-40h], xmm0      // 将临时对象拷贝给局部结构体变量
movups  xmm0, xmmword ptr [eax+10h]
movups  xmmword ptr [ebp-30h], xmm0
movq    xmm0, qword ptr [eax+20h]
movq    qword ptr [ebp-20h], xmm0
push    dword ptr [ebp-1Ch]
lea     eax, [ebp-40h]
push    eax
push    offset aSD_0    ; "%s %d\n"
call    printf

// 如果分析出的函数有以下特征
obj * GetObj(Obj *pRet)
{
    // 功能操作
    ...
    // 将结果拷贝给参数并返回
    memcpy(pRet, xxx, xxx);
    return pRet;
}

// 于是等价于
obj GetObj()
{
    obj = xxx;
    ...
    return obj;
}
```