易逆向分析指南

本文约定:
统一使用:
置入代码 ({ 235, 254, 235, 254, 235, 254 }) 来定位代码

```
0040326C  - EB FE              jmp short 易逆向.0040326C
0040326E  - EB FE              jmp short 易逆向.0040326E
00403270  - EB FE              jmp short 易逆向.00403270
```

EBFE 是死循环指令，程序跑起来后直接暂停就会停在这里。
为了让结构更加清晰，程序内会使用 置入代码 (144) nop 指令
本文的大部分内容适用于易语言 5.x 的版本（一般编译，静态编译），5.x 以下版本没测试过~.~
本人的语文不好，有读不通的地方，还请谅解。
本文仅做抛砖引玉的作用，细节还请自己独立分析。
欢迎各位大侠补充。
作者:无名侠（看雪，精易，鱼 C）
小站:www.pandaos.net
QQ:1447380573
交流群:173836071

流程控制语句篇:

1、计次循环
　　计次循环是易语言的基础流程语句之一。次数从 1 开始。

子程序名	返回值类型	公开	备 注
Fun1	整数型		

变量名	类 型	静态	数组	备 注
x	整数型			
i	整数型			
float	双精度小数型			

```
置入代码 ({ 235, 254, 235, 254, 235, 254 }) ' 方便定位
x = 10
计次循环首 (100, i)
    x = x + i
计次循环尾 ()
返回 (10)
置入代码 ({ 235, 254, 235, 254, 235, 254 })
```

反汇编结果:

```
00403255    55              push ebp
00403256    8BEC            mov ebp,esp
00403258    81EC 20000000   sub esp,0x20
0040325E    C745 FC 0000000>mov dword ptr ss:[ebp-0x4],0x0
00403265    C745 F8 0000000>mov dword ptr ss:[ebp-0x8],0x0
0040326C  - EB FE           jmp short 易逆向.0040326C
0040326E  - EB FE           jmp short 易逆向.0040326E
00403270  - EB FE           jmp short 易逆向.00403270
00403272    C745 FC 0A00000>mov dword ptr ss:[ebp-0x4],0xA
00403279    33C9            xor ecx,ecx                          ; ecx=0
```

```
0040327B    8D45 F8         lea eax,dword ptr ss:[ebp-0x8]
0040327E    8BD8            mov ebx,eax                     ; ebx 指向计次变量地址
00403280    41              inc ecx                         ; 循环量从 1 开始，每次循环都执行这里
00403281    51              push ecx
00403282    53              push ebx
00403283    890B            mov dword ptr ds:[ebx],ecx      ; 给计次变量赋值
00403285    83F9 64         cmp ecx,0x64
00403288    0F8F 24000000   jg 易逆向.004032B2                ; 判断是否超过循环次数
0040328E    DB45 FC         fild dword ptr ss:[ebp-0x4]     ; 循环体
00403291    DD5D F0         fstp qword ptr ss:[ebp-0x10]
00403294    DD45 F0         fld qword ptr ss:[ebp-0x10]
00403297    DB45 F8         fild dword ptr ss:[ebp-0x8]
0040329A    DD5D E8         fstp qword ptr ss:[ebp-0x18]
0040329D    DC45 E8         fadd qword ptr ss:[ebp-0x18]
004032A0    DD5D E0         fstp qword ptr ss:[ebp-0x20]
004032A3    DD45 E0         fld qword ptr ss:[ebp-0x20]
004032A6    E8 83FFFFFF     call 易逆向.0040322E              ; 小数转为整数
004032AB    8945 FC         mov dword ptr ss:[ebp-0x4],eax
004032AE    5B              pop ebx                         ; 0012FC8C
004032AF    59              pop ecx                         ; 0012FC8C
004032B0  ^ EB CE           jmp short 易逆向.00403280         ; 向上跳到循环首。
004032B2    83C4 08         add esp,0x8                     ; 平衡栈
004032B5    B8 0A000000     mov eax,0xA                     ; 返回值
004032BA    E9 06000000     jmp 易逆向.004032C5
004032BF  - EB FE           jmp short 易逆向.004032BF
004032C1  - EB FE           jmp short 易逆向.004032C1
```

```
004032C3  - EB FE          jmp short 易逆向.004032C3
004032C5    8BE5           mov esp,ebp
004032C7    5D             pop ebp                              ; 0012FC8C
```

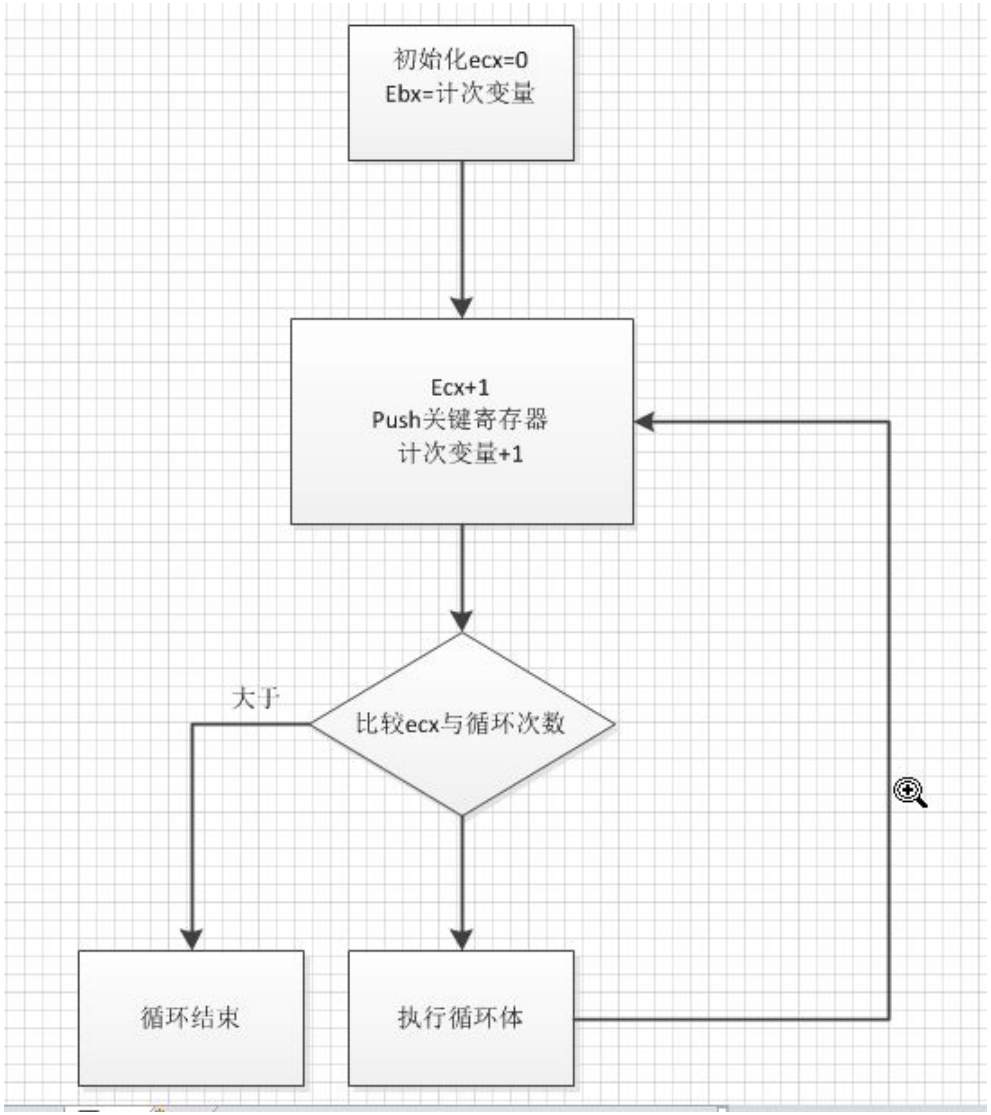
易语言的计次循环语句会以 ecx 作为循环计次。 在进入循环体之前 ecx 被赋值为 1。在进入循环体之前还会对重要寄存器入栈。ebx 寄存器存放了计次变量的指针，上面代码段中 ebx 被赋予了 [ebp-8]的地址，也就是第二个变量。对计次变量赋值以后，就进行循环次数的判断 固定格式如下：

Cmp ecx,XXX

Jg XXX

这段指令之后就是循环体了。在整个流程的最后是一条向上跳的 jmp 指令。重要部分在代码中已经用红色标注。

看看流程图:


初始化ecx=0
Ebx=计次变量
Ecx+1
Push关键寄存器
计次变量+1
大于
比较ecx与循环次数
循环结束
执行循环体

总结形式:符合这个形式就可以认为是一段计次循环代码。

```
s:
inc ecx
push ....
mov dword ptr ds:[ebx],ecx
cmp ecx,目标值
jg END
  ;循环体
pop ....
jmp s ;向上跳
END:
```

2、如果、如果真、判断

先来看看如果语句:
易语言的如果语句类似 C/C++的 if 语句。

子程序名	返回值类型	公开	备 注
Fun2	整数型		

变量名	类 型	静态	数组	备 注
a	整数型			
b	整数型			
c	整数型			

```
置入代码 ({ 235, 254, 235, 254, 235, 254 })
a = 10
如果 (a < 10)
    b = 11
    c = a - b
b = 12 ' else 语句块
    c = a + b
```

反汇编核心段:

```
00403279    C745 FC 0A000000>mov dword ptr ss:[ebp-0x4],0xA          ; 变量 1 赋值为 0xA
00403280    837D FC 0A       cmp dword ptr ss:[ebp-0x4],0xA
00403284    0F8D 2C000000    jge 易逆向.004032B6                      ; 大于等于就跳过代码块
0040328A    C745 F8 0B000000>mov dword ptr ss:[ebp-0x8],0xB          ; 第二个变量赋值 0xB
00403291    DB45 FC          fild dword ptr ss:[ebp-0x4]
00403294    DD5D EC          fstp qword ptr ss:[ebp-0x14]
00403297    DD45 EC          fld qword ptr ss:[ebp-0x14]
0040329A    DB45 F8          fild dword ptr ss:[ebp-0x8]
0040329D    DD5D E4          fstp qword ptr ss:[ebp-0x1C]
004032A0    DC65 E4          fsub qword ptr ss:[ebp-0x1C]
```

```
004032A3    DD5D DC         fstp qword ptr ss:[ebp-0x24]
004032A6    DD45 DC         fld qword ptr ss:[ebp-0x24]
004032A9    E8 80FFFFFF     call 易逆向.0040322E
004032AE    8945 F4         mov dword ptr ss:[ebp-0xC],eax
004032B1    E9 27000000     jmp 易逆向.004032DD
004032B6    C745 F8 0C00000>mov dword ptr ss:[ebp-0x8],0xC          ; else 分支
004032BD    DB45 FC         fild dword ptr ss:[ebp-0x4]
004032C0    DD5D EC         fstp qword ptr ss:[ebp-0x14]
004032C3    DD45 EC         fld qword ptr ss:[ebp-0x14]
004032C6    DB45 F8         fild dword ptr ss:[ebp-0x8]
004032C9    DD5D E4         fstp qword ptr ss:[ebp-0x1C]
004032CC    DC45 E4         fadd qword ptr ss:[ebp-0x1C]
004032CF    DD5D DC         fstp qword ptr ss:[ebp-0x24]
004032D2    DD45 DC         fld qword ptr ss:[ebp-0x24]
004032D5    E8 54FFFFFF     call 易逆向.0040322E
004032DA    8945 F4         mov dword ptr ss:[ebp-0xC],eax
004032DD    8B45 F4         mov eax,dword ptr ss:[ebp-0xC]
```

可以看到，易语言对数值的操作都是用的浮点操作，很反人类吧？
易语言的如果语句类似 C/C++的 if 语句。
形式总结:
影响条件寄存器指令 test,cmp
跳转指令 jge je jne to else
if 代码块
jmp end
else:

else 代码块

end:

Nop

Homework:逆向下面代码：

```
00403255    55              push ebp
00403256    8BEC            mov ebp,esp
00403258    81EC 24000000   sub esp,0x24
0040325E    C745 FC 0000000>mov dword ptr ss:[ebp-0x4],0x0
00403265    C745 F8 0000000>mov dword ptr ss:[ebp-0x8],0x0
0040326C    C745 F4 0000000>mov dword ptr ss:[ebp-0xC],0x0
00403273  - EB FE           jmp short 易逆向.00403273
00403275  - EB FE           jmp short 易逆向.00403275
00403277  - EB FE           jmp short 易逆向.00403277
00403279    C745 FC 0A00000>mov dword ptr ss:[ebp-0x4],0xA
00403280    837D FC 0A      cmp dword ptr ss:[ebp-0x4],0xA
00403284    0F8D 2C000000   jge 易逆向.004032B6
0040328A    C745 F8 0B00000>mov dword ptr ss:[ebp-0x8],0xB
00403291    DB45 FC         fild dword ptr ss:[ebp-0x4]
00403294    DD5D EC         fstp qword ptr ss:[ebp-0x14]
00403297    DD45 EC         fld qword ptr ss:[ebp-0x14]
0040329A    DB45 F8         fild dword ptr ss:[ebp-0x8]
0040329D    DD5D E4         fstp qword ptr ss:[ebp-0x1C]
004032A0    DC65 E4         fsub qword ptr ss:[ebp-0x1C]
004032A3    DD5D DC         fstp qword ptr ss:[ebp-0x24]
004032A6    DD45 DC         fld qword ptr ss:[ebp-0x24]
```

```
004032A9    E8 80FFFFFF      call 易逆向.0040322E
004032AE    8945 F4          mov dword ptr ss:[ebp-0xC],eax
004032B1    E9 27000000      jmp 易逆向.004032DD
004032B6    C745 F8 0C00000>mov dword ptr ss:[ebp-0x8],0xC
004032BD    DB45 FC          fild dword ptr ss:[ebp-0x4]
004032C0    DD5D EC          fstp qword ptr ss:[ebp-0x14]
004032C3    DD45 EC          fld qword ptr ss:[ebp-0x14]
004032C6    DB45 F8          fild dword ptr ss:[ebp-0x8]
004032C9    DD5D E4          fstp qword ptr ss:[ebp-0x1C]
004032CC    DC45 E4          fadd qword ptr ss:[ebp-0x1C]
004032CF    DD5D DC          fstp qword ptr ss:[ebp-0x24]
004032D2    DD45 DC          fld qword ptr ss:[ebp-0x24]
004032D5    E8 54FFFFFF      call 易逆向.0040322E
004032DA    8945 F4          mov dword ptr ss:[ebp-0xC],eax
004032DD    837D F4 14       cmp dword ptr ss:[ebp-0xC],0x14
004032E1    0F85 0C000000    jnz 易逆向.004032F3
004032E7    C745 FC 0100000>mov dword ptr ss:[ebp-0x4],0x1
004032EE    E9 07000000      jmp 易逆向.004032FA
004032F3    C745 F8 0300000>mov dword ptr ss:[ebp-0x8],0x3
004032FA    8B45 F4          mov eax,dword ptr ss:[ebp-0xC]
004032FD    E9 00000000      jmp 易逆向.00403302
00403302    8BE5             mov esp,ebp
00403304    5D               pop ebp
00403305    C3               retn
```

判断语句：

```
a = 10
判断 (a < 10)
    b = 11
    c = a - b
    置入代码 (144)
    b = 12 ' else 语句块
    c = a + b
    置入代码 (144)
    判断 (a = 1)
        c = 1
        判断 (a = 2)
            c = 2
            判断 (a = 3)
                c = 3
                |
    判断 (c = 20)
        a = 1
        b = 3
        判断 (c = 21)
            a = 2
            判断 (c = 22)
                a = 3
                a = 4
返回 (5)
```

反汇编结果:

```
004032DA    8945 F4          mov dword ptr ss:[ebp-0xC],eax
004032DD    837D FC 01       cmp dword ptr ss:[ebp-0x4],0x1
004032E1    0F85 0C000000    jnz 易逆向.004032F3
004032E7    C745 F4 0100000>mov dword ptr ss:[ebp-0xC],0x1
004032EE    E9 27000000      jmp 易逆向.0040331A
004032F3    837D FC 02       cmp dword ptr ss:[ebp-0x4],0x2
004032F7    0F85 0C000000    jnz 易逆向.00403309
004032FD    C745 F4 0200000>mov dword ptr ss:[ebp-0xC],0x2
00403304    E9 11000000      jmp 易逆向.0040331A
00403309    837D FC 03       cmp dword ptr ss:[ebp-0x4],0x3
0040330D    0F85 07000000    jnz 易逆向.0040331A
00403313    C745 F4 0300000>mov dword ptr ss:[ebp-0xC],0x3
0040331A    837D F4 14       cmp dword ptr ss:[ebp-0xC],0x14
0040331E    0F85 13000000    jnz 易逆向.00403337
00403324    C745 FC 0100000>mov dword ptr ss:[ebp-0x4],0x1
0040332B    C745 F8 0300000>mov dword ptr ss:[ebp-0x8],0x3
00403332    E9 33000000      jmp 易逆向.0040336A
00403337    837D F4 15       cmp dword ptr ss:[ebp-0xC],0x15
0040333B    0F85 0C000000    jnz 易逆向.0040334D
00403341    C745 FC 0200000>mov dword ptr ss:[ebp-0x4],0x2
00403348    E9 1D000000      jmp 易逆向.0040336A
0040334D    837D F4 16       cmp dword ptr ss:[ebp-0xC],0x16
00403351    0F85 0C000000    jnz 易逆向.00403363
00403357    C745 FC 0300000>mov dword ptr ss:[ebp-0x4],0x3
0040335E    E9 07000000      jmp 易逆向.0040336A
```

```
00403363     C745 FC 0400000>mov dword ptr ss:[ebp-0x4],0x4
0040336A     B8 05000000     mov eax,0x5
```

可以看出，易语言的判断和如果是差不多的，不知道那种类 C 的 Switch 结构是怎么搞出来的。

3、判断循环首 与 循环判断首

判断循环首是先判断再循环。

子程序名	返回值类型	公开	备 注
Fun3			

变量名	类 型	静态	数组	备 注
i	整数型			

```
置入代码 ({ 235, 254, 235, 254, 235, 254 })
i = 10
判断循环首 (i < 20)
    i = i + 1
判断循环尾 ()
```

```
00403244     C745 FC 0A00000>mov dword ptr ss:[ebp-0x4],0xA          ; 变量 i 赋值
0040324B     837D FC 14      cmp dword ptr ss:[ebp-0x4],0x14         ; 变量 i 的值与 0x14 比较
0040324F     0F8D 05000000   jge 易逆向.0040325A                       ; 大于则跳到循环尾
00403255     FF45 FC         inc dword ptr ss:[ebp-0x4]              ; 循环体
00403258   ^ EB F1           jmp short 易逆向.0040324B
0040325A     8BE5            mov esp,ebp
```

形式总结:
Start:
条件测试指令....
条件跳转 to end
循环体....
Jmp Start 向上跳转
end
符合上面形式的代码就可以认为是一段判断循环首代码，有点类似于 C/C++的 while 语句。

关于 跳出循环 () 与 到循环尾 () 流程控制语句:

子程序名	返回值类型	公开	备 注
Fun3			

变量名	类 型	静态	数组	备 注
i	整数型			

```
置入代码 ({ 235, 254, 235, 254, 235, 254 })
i = 10
判断循环首 (i < 20)
    i = i + 1
    如果真 (i = 15)
        跳出循环 ()
    如果真 (i = 16)
        到循环尾 ()

判断循环尾 ()
```

```
00403244    C745 FC 0A000000>mov dword ptr ss:[ebp-0x4],0xA
0040324B    837D FC 14       cmp dword ptr ss:[ebp-0x4],0x14
0040324F    0F8D 23000000    jge 易逆向.00403278
00403255    FF45 FC          inc dword ptr ss:[ebp-0x4]
00403258    837D FC 0F       cmp dword ptr ss:[ebp-0x4],0xF
0040325C    0F85 05000000    jnz 易逆向.00403267
00403262    E9 11000000      jmp 易逆向.00403278                    ; break  跳出循环
00403267    837D FC 10       cmp dword ptr ss:[ebp-0x4],0x10
0040326B    0F85 05000000    jnz 易逆向.00403276                    ; 易语言的优化做的很不好
00403271  ^ E9 D5FFFFFF      jmp 易逆向.0040324B                    ; continue 到循环尾
00403276  ^ EB D3            jmp short 易逆向.0040324B
00403278    8BE5             mov esp,ebp
```

跳出循环实际上就相当于 C 语言的 break 语句，无条件 jmp 到循环体外。 到循环尾() 相当于 C 语言的 continue 语句。按照反汇编结果中显示的来理解，到循环尾()实际上是“到循环头()”

循环判断首:

先 循环 再 判断。

子程序名	返回值类型	公开	备 注
Fun3			

变量名	类 型	静态	数组	备 注
i	整数型			

```
00403244    C745 FC 0A000000>mov dword ptr ss:[ebp-0x4],0xA          ; 给变量 i 赋值 i=0xA
0040324B    FF45 FC          inc dword ptr ss:[ebp-0x4]              ; 先执行循环体
0040324E    837D FC 0F       cmp dword ptr ss:[ebp-0x4],0xF
00403252    0F85 05000000    jnz 易逆向.0040325D
00403258    E9 15000000      jmp 易逆向.00403272                       ; 跳出循环
0040325D    837D FC 10       cmp dword ptr ss:[ebp-0x4],0x10
00403261    0F85 05000000    jnz 易逆向.0040326C                       ; 到循环尾
00403267    E9 00000000      jmp 易逆向.0040326C
0040326C    837D FC 14       cmp dword ptr ss:[ebp-0x4],0x14
00403270  ^ 7C D9            jl short 易逆向.0040324B                  ; 满足条件则向上跳
00403272    8BE5             mov esp,ebp
```

先循环也就是先执行循环体，然后才是对条件进行判断，满足条件则向上跳转。
形式总结:
Start:
循环体.....
条件测试指令....
条件跳转 to Start

易语言的流程控制已经介绍的差不多了，谈谈我的感受。易语言的目标文件优化做的很不好如下面代码片段：

```
00403267    837D FC 10       cmp dword ptr ss:[ebp-0x4],0x10
0040326B    0F85 05000000    jnz 易逆向.00403276                       ; 易语言的优化做的很不好
00403271  ^ E9 D5FFFFFF      jmp 易逆向.0040324B                       ; continue 到循环尾
00403276  ^ EB D3            jmp short 易逆向.0040324B
```

这 4 条指令很明显可以直接合并为一条跳转。

整数运算代码片:实现 x=x+i 的功能 x 与 i 都为整数型。

```
0040328E    DB45 FC         fild dword ptr ss:[ebp-0x4]             ; 循环体
00403291    DD5D F0         fstp qword ptr ss:[ebp-0x10]
00403294    DD45 F0         fld qword ptr ss:[ebp-0x10]
00403297    DB45 F8         fild dword ptr ss:[ebp-0x8]
0040329A    DD5D E8         fstp qword ptr ss:[ebp-0x18]
0040329D    DC45 E8         fadd qword ptr ss:[ebp-0x18]
004032A0    DD5D E0         fstp qword ptr ss:[ebp-0x20]
004032A3    DD45 E0         fld qword ptr ss:[ebp-0x20]
004032A6    E8 83FFFFFF     call 易逆向.0040322E                    ; 小数转为整数
004032AB    8945 FC         mov dword ptr ss:[ebp-0x4],eax
```

明明几条 ADD 指令就能搞定的东西非要弄这么复杂.....无力吐槽中。

易语言函数框架:

易语言的主要调用方式是 stdcall。
_stdcall 的规则是参数采用从右到左的压栈方式，由调用者完成压栈操作 ，传送参数的内存栈由调用者维护。Windows 的大部分 API 调用都是采用的 _stdcall ，这一点易语言的老大是做的很好的。

子程序名	返回值类型	公开	备 注		
fun4	整数型		非导出函数		
参数名	类 型	参考	可空	数组	备 注
i	整数型				

```
i = 5
fun5 (i)
返回 (i)
```

子程序名	返回值类型	公开	备 注		
fun5	整数型	✔	导出函数		
参数名	类 型	参考	可空	数组	备 注
i					

```
返回 (i + 2)
```

```
004032C4    55              push ebp
004032C5    8BEC            mov ebp,esp                          ; 创建函数框架
004032C7    81EC 10000000    sub esp,0x10                         ; 为局部变量开辟栈空间
004032CD    DB45 08         fild dword ptr ss:[ebp+0x8]           ; 函数体
004032D0    DD5D F8         fstp qword ptr ss:[ebp-0x8]
004032D3    DD45 F8         fld qword ptr ss:[ebp-0x8]
004032D6    DC05 CB304000    fadd qword ptr ds:[0x4030CB]
004032DC    DD5D F0         fstp qword ptr ss:[ebp-0x10]
004032DF    DD45 F0         fld qword ptr ss:[ebp-0x10]
004032E2    E8 B6FFFFFF      call 易逆向.0040329D                 ; 小数转为整数
```

```
004032E7    E9 00000000     jmp 易逆向.004032EC                    ; 反人类的 jmp
004032EC    8BE5            mov esp,ebp                            ; 平衡栈
004032EE    5D              pop ebp                                ; 0
004032EF    C2 0400         retn 0x4                               ; 返回,参数的大小为 0x4
```

参数：[ebp+4+参数偏移]
局部变量:[ebp-4-参数偏移]

注意返回值是在 eax 中的。
函数框架：

```
Push ebp
Mov ebp,esp
```

函数体

```
Mov esp,ebp
Pop ebp
Retn x
```

函数头是很好识别的，在 OD 中可以快速定位。易语言的优化做的很差，不必担心被”优化”的函数体。
调用方：

```
Push XXX
Call XXX
```

易语言运算

程序名	返回值类型	公开	备 注
fun6	整数型	✔	

变量名	类 型	静态	数组	备 注
a	整数型			
b	整数型			
c	整数型			
d	整数型			

```
a = 10
b = 20
c = a + b
a = b - c
b = c × 2
d = a ÷ 2
返回 (a % 3)
```

C = a+b 反汇编结果:

```
004032DA    DB45 FC        fild dword ptr ss:[ebp-0x4]            ; 这个就是 a
004032DD    DD5D E8        fstp qword ptr ss:[ebp-0x18]          ; 转换结果放临时变量
004032E0    DD45 E8        fld qword ptr ss:[ebp-0x18]
004032E3    DB45 F8        fild dword ptr ss:[ebp-0x8]            ; 这个就是 b
004032E6    DD5D E0        fstp qword ptr ss:[ebp-0x20]          ; 把转换后的浮点数放到临时变量
004032E9    DC45 E0        fadd qword ptr ss:[ebp-0x20]          ;ST0 + [ebp-0x20]的值
004032EC    DD5D D8        fstp qword ptr ss:[ebp-0x28]
```

```
004032EF    DD45 D8         fld qword ptr ss:[ebp-0x28]
004032F2    E8 89FFFFFF     call 易逆向.00403280                    ; 把浮点数转为整数 参数 ST0 寄存器的值
004032F7    8945 F4         mov dword ptr ss:[ebp-0xC],eax          ; 保存运算结果
```

易语言对所以的运算都用的是浮点运算，对浮点指令不熟悉的同学可以去看雪找一些相关资料。
我们只看关键指令，fadd，只要有这条指令就可以认为是一段相加语句。 不必担心几种运算放在浮点寄存器里一起搞
因为易语言的优化是做的很差的！
执行到最后，程序通过调用 **call** 易逆向.00403280 将浮点寄存器 ST0 的值转换为整数。
有兴趣的同学可以阅读一下：

```
00403280    55              push ebp
00403281    8BEC            mov ebp,esp
00403283    83C4 F4         add esp,-0xC
00403286    D97D FE         fstcw word ptr ss:[ebp-0x2]
00403289    66:8B45 FE      mov ax,word ptr ss:[ebp-0x2]
0040328D    80CC 0C         or ah,0xC
00403290    66:8945 FC      mov word ptr ss:[ebp-0x4],ax
00403294    D96D FC         fldcw word ptr ss:[ebp-0x4]
00403297    DF7D F4         fistp qword ptr ss:[ebp-0xC]
0040329A    D96D FE         fldcw word ptr ss:[ebp-0x2]
0040329D    8B45 F4         mov eax,dword ptr ss:[ebp-0xC]
004032A0    8B55 F8         mov edx,dword ptr ss:[ebp-0x8]
004032A3    8BE5            mov esp,ebp
004032A5    5D              pop ebp
004032A6    C3              retn
```

a = b - c 反汇编结果：

```
004032FA    DB45 F8         fild dword ptr ss:[ebp-0x8]
004032FD    DD5D E8         fstp qword ptr ss:[ebp-0x18]
00403300    DD45 E8         fld qword ptr ss:[ebp-0x18]
00403303    DB45 F4         fild dword ptr ss:[ebp-0xC]
00403306    DD5D E0         fstp qword ptr ss:[ebp-0x20]
00403309    DC65 E0         fsub qword ptr ss:[ebp-0x20]
0040330C    DD5D D8         fstp qword ptr ss:[ebp-0x28]
0040330F    DD45 D8         fld qword ptr ss:[ebp-0x28]
00403312    E8 69FFFFFF     call 易逆向.00403280
00403317    8945 FC         mov dword ptr ss:[ebp-0x4],eax
```

有一条关键指令：fsub 就可以认为这是一段减法语句。

b = c × 2 反汇编结果：

```
0040331A    DB45 F4         fild dword ptr ss:[ebp-0xC]
0040331D    DD5D E8         fstp qword ptr ss:[ebp-0x18]
00403320    DD45 E8         fld qword ptr ss:[ebp-0x18]
00403323    DC0D CB304000   fmul qword ptr ds:[0x4030CB]  这里是一个常量值
00403329    DD5D E0         fstp qword ptr ss:[ebp-0x20]
0040332C    DD45 E0         fld qword ptr ss:[ebp-0x20]
0040332F    E8 4CFFFFFF     call 易逆向.00403280
00403334    8945 F8         mov dword ptr ss:[ebp-0x8],eax
```

Fmul 是乘法指令

d = a ÷ 2 反汇编结果:

```
00403337    DB45 FC         fild dword ptr ss:[ebp-0x4]
0040333A    DD5D E8         fstp qword ptr ss:[ebp-0x18]
0040333D    DD45 E8         fld qword ptr ss:[ebp-0x18]
00403340    DC35 CB304000    fdiv qword ptr ds:[0x4030CB]
00403346    DD5D E0         fstp qword ptr ss:[ebp-0x20]
00403349    DD45 E0         fld qword ptr ss:[ebp-0x20]
0040334C    E8 2FFFFFFF     call 易逆向.00403280
00403351    8945 F0         mov dword ptr ss:[ebp-0x10],eax
```

关键指令:fdiv