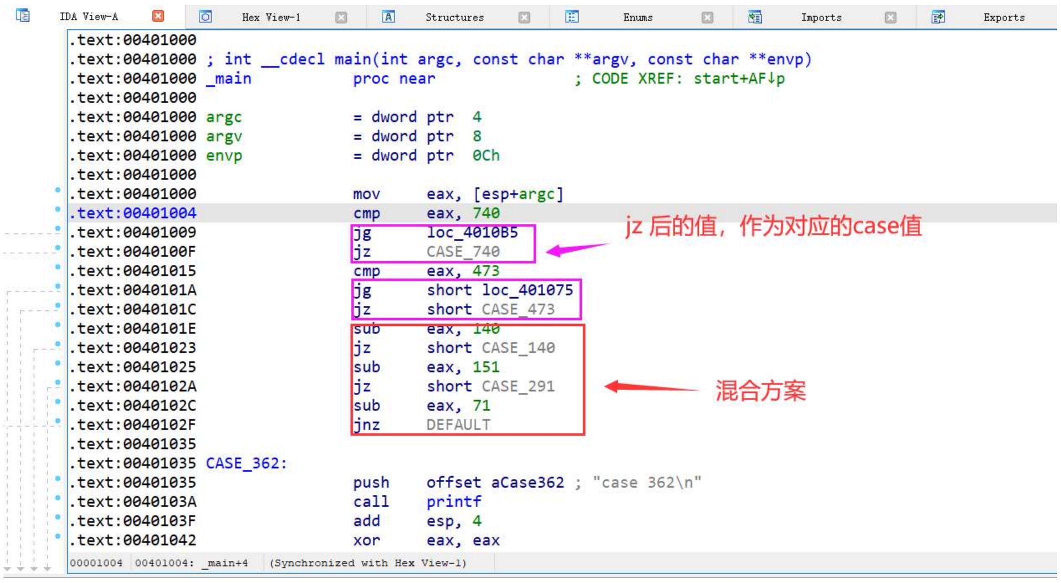
2021/03/17_x86逆向_第10课_switch-case情况4、循环结构

笔记本： x86逆向-C
创建时间： 2021/3/17 星期三 9:58
作者： ileemi

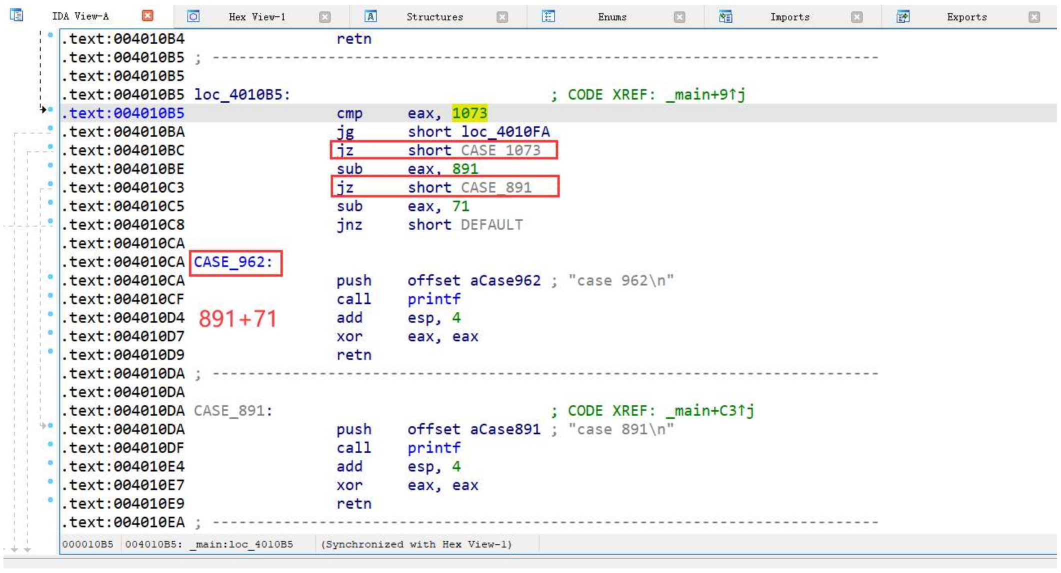
switch-case

情况4（降低判定树的高度）： 最大 case 值和最小 case 值差值大于255时，会出现两个跳转在一起（jg、jz）的情况。编译器在编译时，会将 case 值进行从小到大排序，将每一个 case 值作为一个节点，从这些节点中找到一个中间值作为根节点，以此形成一课平衡二叉树，以每个节点为判定值，大于和小于关系分别对应左子树和右子树。

示例：

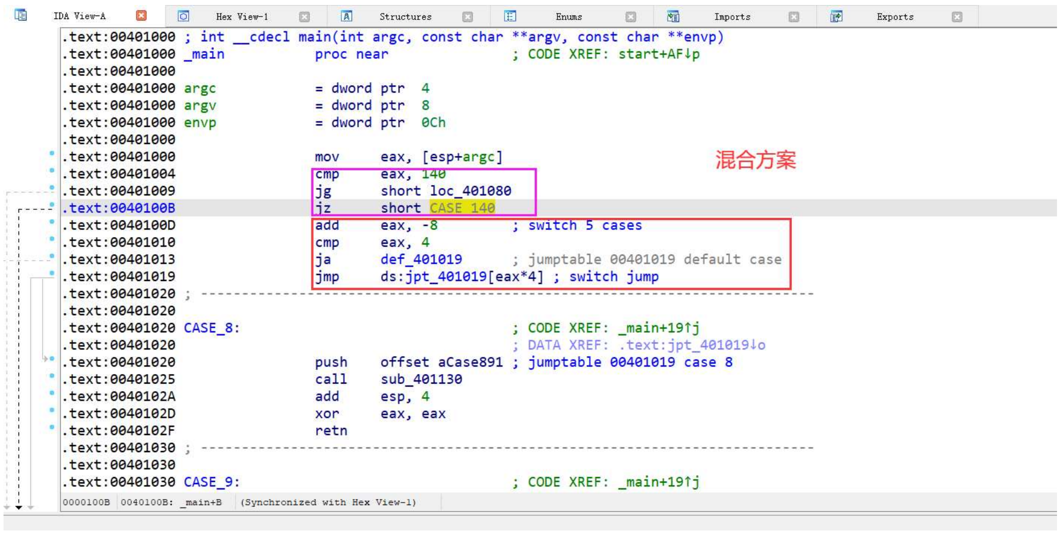


```
.text:00401000
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main           proc near               ; CODE XREF: start+AF↓p
.text:00401000
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 mov     eax, [esp+argc]
.text:00401004                 cmp     eax, 740
.text:00401009                 jg      loc_4010B5
.text:0040100F                 jz      CASE_740
.text:00401015                 cmp     eax, 473
.text:0040101A                 jg      short loc_401075
.text:0040101C                 jz      short CASE_473
.text:0040101E                 sub     eax, 140
.text:00401023                 jz      short CASE_140
.text:00401025                 sub     eax, 151
.text:0040102A                 jz      short CASE_291
.text:0040102C                 sub     eax, 71
.text:0040102F                 jnz     DEFAULT
.text:00401035
.text:00401035 CASE_362:
.text:00401035                 push    offset aCase362 ; "case 362\n"
.text:0040103A                 call    printf
.text:0040103F                 add     esp, 4
.text:00401042                 xor     eax, eax
00001004 00401004: _main+4  (Synchronized with Hex View-1)
```



```
.text:004010B4                 retn
.text:004010B5 ; ---------------------------------------------------------------------------
.text:004010B5
.text:004010B5 loc_4010B5:                             ; CODE XREF: _main+9↑j
.text:004010B5                 cmp     eax, 1073
.text:004010BA                 jg      short loc_4010FA
.text:004010BC                 jz      short CASE_1073
.text:004010BE                 sub     eax, 891
.text:004010C3                 jz      short CASE_891
.text:004010C5                 sub     eax, 71
.text:004010C8                 jnz     short DEFAULT
.text:004010CA
.text:004010CA CASE_962:
.text:004010CA                 push    offset aCase962 ; "case 962\n"
.text:004010CF                 call    printf
.text:004010D4                 add     esp, 4
.text:004010D7                 xor     eax, eax
.text:004010D9                 retn
.text:004010DA ; ---------------------------------------------------------------------------
.text:004010DA
.text:004010DA CASE_891:                               ; CODE XREF: _main+C3↑j
.text:004010DA                 push    offset aCase891 ; "case 891\n"
.text:004010DF                 call    printf
.text:004010E4                 add     esp, 4
.text:004010E7                 xor     eax, eax
.text:004010E9                 retn
.text:004010EA ; ---------------------------------------------------------------------------
000010B5 004010B5: _main:loc_4010B5    (Synchronized with Hex View-1)
```

该情况存在混合方案（取决与 case 值之间是否连续以及值之间的差值大小），如下图所示：



```
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main           proc near               ; CODE XREF: start+AF↓p
.text:00401000
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 mov     eax, [esp+argc]
.text:00401004                 cmp     eax, 140
.text:00401009                 jg      short loc_401080
.text:0040100B                 jz      short CASE_140
.text:0040100D                 add     eax, -8         ; switch 5 cases
.text:00401010                 cmp     eax, 4
.text:00401013                 ja      def_401019      ; jumptable 00401019 default case
.text:00401019                 jmp     ds:jpt_401019[eax*4] ; switch jump
.text:00401020 ; ---------------------------------------------------------------------------
.text:00401020
.text:00401020 CASE_8:                                 ; CODE XREF: _main+19↑j
.text:00401020                                         ; DATA XREF: .text:jpt_401019↓o
.text:00401020                 push    offset aCase891 ; jumptable 00401019 case 8
.text:00401025                 call    sub_401130
.text:0040102A                 add     esp, 4
.text:0040102D                 xor     eax, eax
.text:0040102F                 retn
.text:00401030 ; ---------------------------------------------------------------------------
.text:00401030
.text:00401030 CASE_9:                                 ; CODE XREF: _main+19↑j
0000100B 0040100B: _main+B    (Synchronized with Hex View-1)
```

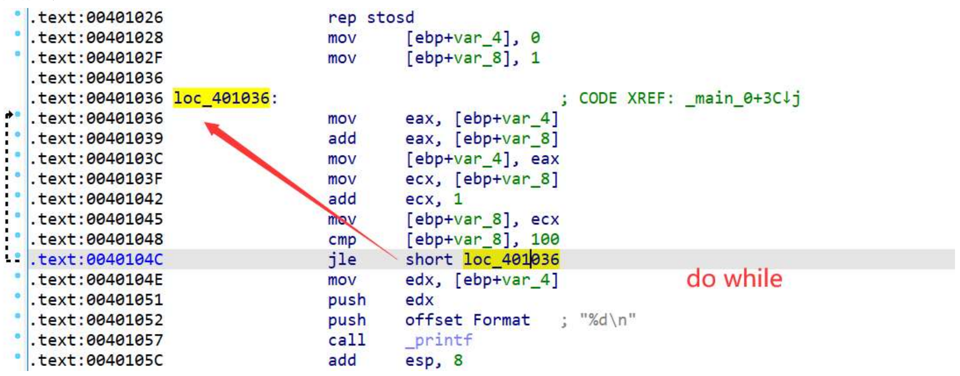
循环结构

do-while

> 先执行循环体，后比较判断。3种循环中效率最高的，还原代码时，条件按照正条件进行还原。

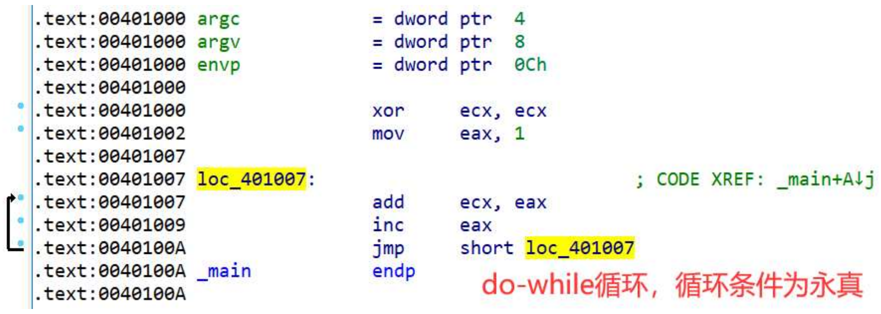
示例：

- 正常情况



```
.text:00401026                 rep stosd
.text:00401028                 mov     [ebp+var_4], 0
.text:0040102F                 mov     [ebp+var_8], 1
.text:00401036
.text:00401036 loc_401036:                             ; CODE XREF: _main_0+3C↓j
.text:00401036                 mov     eax, [ebp+var_4]
.text:00401039                 add     eax, [ebp+var_8]
.text:0040103C                 mov     [ebp+var_4], eax
.text:0040103F                 mov     ecx, [ebp+var_8]
.text:00401042                 add     ecx, 1
.text:00401045                 mov     [ebp+var_8], ecx
.text:00401048                 cmp     [ebp+var_8], 100
.text:0040104C                 jle     short loc_401036
.text:0040104E                 mov     edx, [ebp+var_4]
.text:00401051                 push    edx
.text:00401052                 push    offset Format   ; "%d\n"
.text:00401057                 call    _printf
.text:0040105C                 add     esp, 8
```

- 永真循环（死循环）



```
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 xor     ecx, ecx
.text:00401002                 mov     eax, 1
.text:00401007
.text:00401007 loc_401007:                             ; CODE XREF: _main+A↓j
.text:00401007                 add     ecx, eax
.text:00401009                 inc     eax
.text:0040100A                 jmp     short loc_401007
.text:0040100A _main           endp
.text:0040100A
```

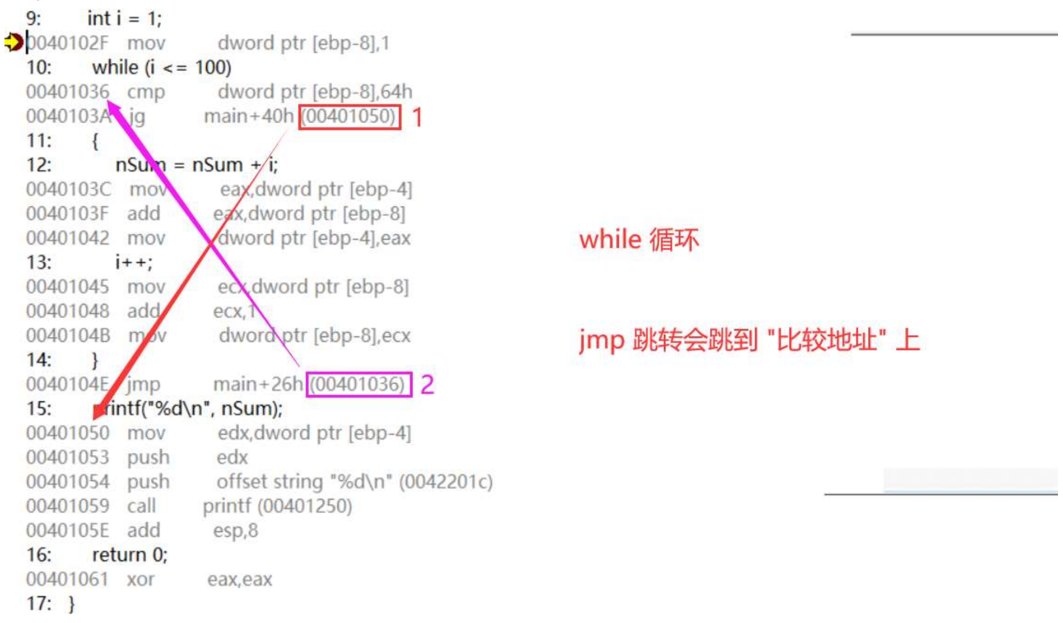
> 代码定式：

```
DO_BEGIN:
  ... // 循环语句块
  jxx DO_BEGIN // 向上跳转
DO_END:
```

while

> 先比较判断，后执行循环体。反汇编还原代码时需要反条件还原，和流水线方向不一致。

示例：



```
9:      int i = 1;
0040102F   mov         dword ptr [ebp-8],1
10:     while (i <= 100)
00401036   cmp         dword ptr [ebp-8],64h
0040103A   jg          main+40h (00401050)
11:     {
12:         nSum = nSum + i;
0040103C   mov         eax,dword ptr [ebp-4]
0040103F   add         eax,dword ptr [ebp-8]
00401042   mov         dword ptr [ebp-4],eax
13:         i++;
00401045   mov         ecx,dword ptr [ebp-8]
00401048   add         ecx,1
0040104B   mov         dword ptr [ebp-8],ecx
14:     }
0040104E   jmp         main+26h (00401036)
15:     printf("%d\n", nSum);
00401050   mov         edx,dword ptr [ebp-4]
00401053   push        edx
00401054   push        offset string "%d\n" (0042201c)
00401059   call        printf (00401250)
0040105E   add         esp,8
16:     return 0;
00401061   xor         eax,eax
17: }
```

> 代码定式:

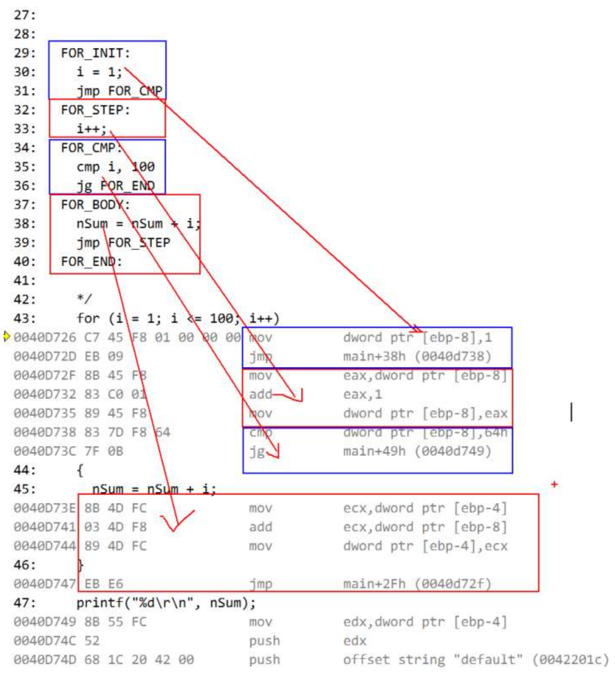
```
WHILE_BEGIN:
   jxx WHILE_END
   ... // 循环语句块
   jmp DO_BEGIN // 向上跳转
WHILE_END:
```

for

> 先初始化，再比较判断，后执行循环体。反汇编还原代码时需要反条件还原，和流水线方向不一致。

伪代码如下:

```
FOR_INIT:
   i = 1;
   jmp FOR_CMP
FOR_STEP:
   i++;
FOR_CMP:
   cmp i, 100
   jg FOR_END
FOR_BODY:
   nSum = nSum + i;
   jmp FOR_STEP
FOR_END:
```



```
27:
28:
29:   FOR_INIT:
30:     i = 1;
31:     jmp FOR_CMP
32:   FOR_STEP:
33:     i++;
34:   FOR_CMP:
35:     cmp i, 100
36:     jg FOR_END
37:   FOR_BODY:
38:     nSum = nSum + i;
39:     jmp FOR_STEP
40:   FOR_END:
41:
42:     */
43:     for (i = 1; i <= 100; i++)
0040D726 C7 45 F8 01 00 00 00 mov         dword ptr [ebp-8],1
0040D72D EB 09                jmp         main+38h (0040d738)
0040D72F 8B 45 F8             mov         eax,dword ptr [ebp-8]
0040D732 83 C0 01             add         eax,1
0040D735 89 45 F8             mov         dword ptr [ebp-8],eax
0040D738 83 7D F8 64          cmp         dword ptr [ebp-8],64h
0040D73C 7F 0B                jg          main+49h (0040d749)
44:     {
45:       nSum = nSum + i;
0040D73E 8B 4D FC             mov         ecx,dword ptr [ebp-4]
0040D741 03 4D F8             add         ecx,dword ptr [ebp-8]
0040D744 89 4D FC             mov         dword ptr [ebp-4],ecx
46:     }
0040D747 EB E6                jmp         main+2Fh (0040d72f)
47:     printf("%d\r\n", nSum);
0040D749 8B 55 FC             mov         edx,dword ptr [ebp-4]
0040D74C 52                   push        edx
0040D74D 68 1C 20 42 00       push        offset string "default" (0042201c)
```

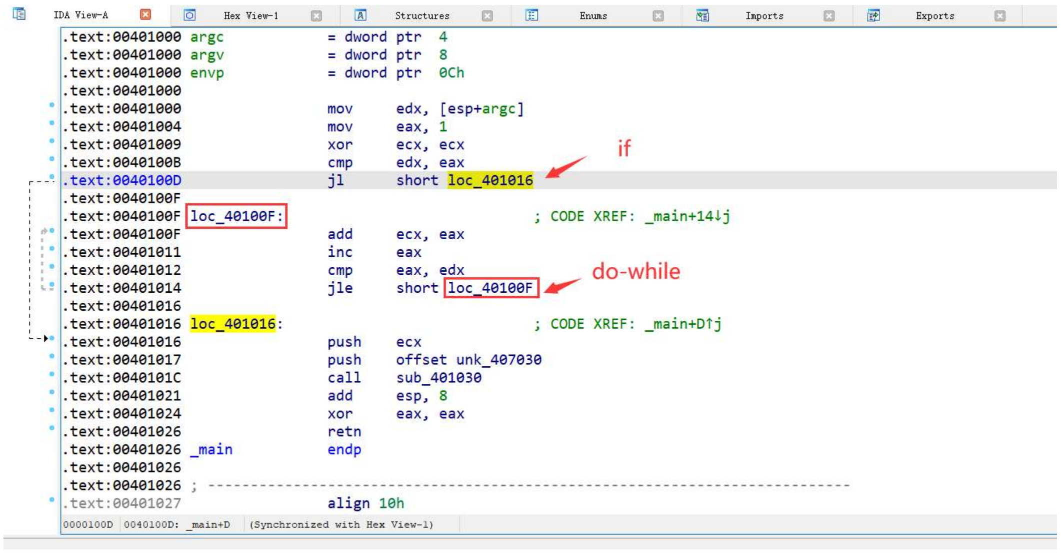
汇编代码定式:

```
  mov mem/reg, xxx // 赋初值
  jmp FOR_CMP // 跳转到循环条件判定部分
FOR_STEP: // 步长计算部分
// 修改循环变量Step
  mov reg, Step
  // 修改循变量的计算过程，在实际分析中，视算法不同而不同
  add reg, xxx
  mov Step, eax
FOR_CMP: // 循环条件判定部分
  mov eax. dword ptr Step
  // 判断循环变量和循环终止条件StepEnd的关系，
  // 满足条件则退出 for 循环
  cmp ecx, StepEnd
  jxx FOR_END // 条件成立，循环结束
  ...
  jmp FOR_STEP // 向上跳转，修改流程回到步长计算部分
FOR_END:
```

> 上面的三种循环，定式时没有Debug版的，在 Release 版中，三种循环最终都会优化成 "do-while" 循环结构。

比较条件表达式中有常量时，就不满足常量折叠，所以生成的汇编代码中在do-while外有一层判断语句（if），结构如下所示：

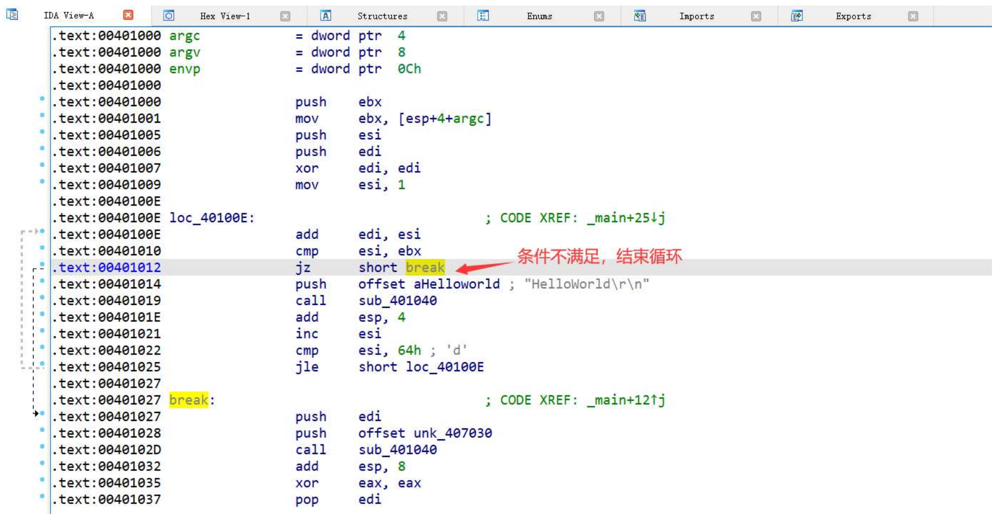
```
for(i = 1; i <= argc; i++)
{
  nSum = nSum + i;
}
// 伪代码
if (i <= argc)
{
  do
  {
    nSum = nSum + i;
    i++;
  }while(i <= argc)
}
```



```
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 mov     edx, [esp+argc]
.text:00401004                 mov     eax, 1
.text:00401009                 xor     ecx, ecx
.text:0040100B                 cmp     edx, eax
.text:0040100D                 jl      short loc_401016
.text:0040100F
.text:0040100F loc_40100F:                             ; CODE XREF: _main+14↓j
.text:0040100F                 add     ecx, eax
.text:00401011                 inc     eax
.text:00401012                 cmp     eax, edx
.text:00401014                 jle     short loc_40100F
.text:00401016
.text:00401016 loc_401016:                             ; CODE XREF: _main+D↑j
.text:00401016                 push    ecx
.text:00401017                 push    offset unk_407030
.text:0040101C                 call    sub_401030
.text:00401021                 add     esp, 8
.text:00401024                 xor     eax, eax
.text:00401026                 retn
.text:00401026 _main           endp
.text:00401026
.text:00401026 ; ---------------------------------------------------------------------------
.text:00401027                 align 10h
0000100D 0040100D: _main+D  (Synchronized with Hex View-1)
```

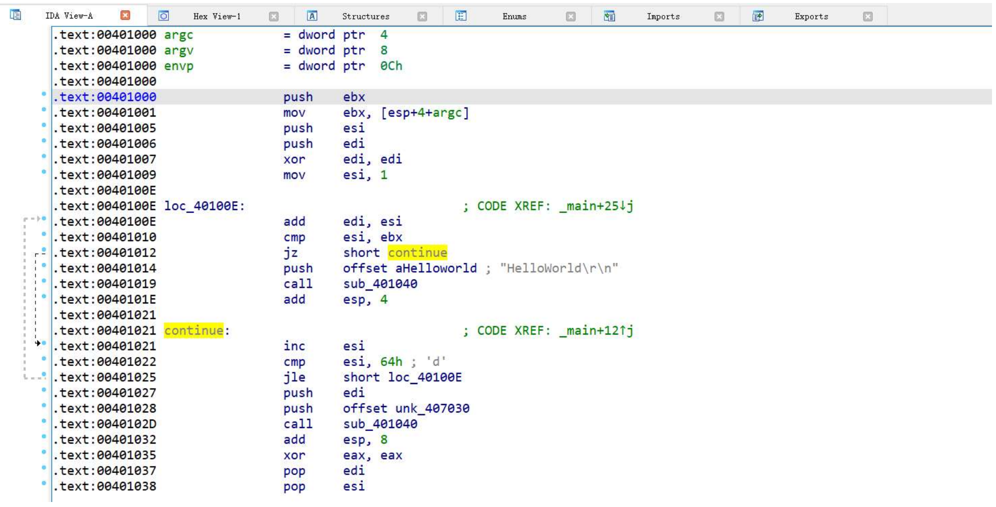
在循环中根据跳转处的上下文汇编代码可以分辨出break、continue：

- break（跳转到循环结束位置）



```
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 push    ebx
.text:00401001                 mov     ebx, [esp+4+argc]
.text:00401005                 push    esi
.text:00401006                 push    edi
.text:00401007                 xor     edi, edi
.text:00401009                 mov     esi, 1
.text:0040100E
.text:0040100E loc_40100E:                             ; CODE XREF: _main+25↓j
.text:0040100E                 add     edi, esi
.text:00401010                 cmp     esi, ebx
.text:00401012                 jz      short break
.text:00401014                 push    offset aHelloworld ; "HelloWorld\r\n"
.text:00401019                 call    sub_401040
.text:0040101E                 add     esp, 4
.text:00401021                 inc     esi
.text:00401022                 cmp     esi, 64h ; 'd'
.text:00401025                 jle     short loc_40100E
.text:00401027
.text:00401027 break:                                  ; CODE XREF: _main+12↑j
.text:00401027                 push    edi
.text:00401028                 push    offset unk_407030
.text:0040102D                 call    sub_401040
.text:00401032                 add     esp, 8
.text:00401035                 xor     eax, eax
.text:00401037                 pop     edi
```

- continue（跳过后面需要执行的语句）



```
.text:00401000 argc            = dword ptr  4
.text:00401000 argv            = dword ptr  8
.text:00401000 envp            = dword ptr  0Ch
.text:00401000
.text:00401000                 push    ebx
.text:00401001                 mov     ebx, [esp+4+argc]
.text:00401005                 push    esi
.text:00401006                 push    edi
.text:00401007                 xor     edi, edi
.text:00401009                 mov     esi, 1
.text:0040100E
.text:0040100E loc_40100E:                             ; CODE XREF: _main+25↓j
.text:0040100E                 add     edi, esi
.text:00401010                 cmp     esi, ebx
.text:00401012                 jz      short continue
.text:00401014                 push    offset aHelloworld ; "HelloWorld\r\n"
.text:00401019                 call    sub_401040
.text:0040101E                 add     esp, 4
.text:00401021
.text:00401021 continue:                               ; CODE XREF: _main+12↑j
.text:00401021                 inc     esi
.text:00401022                 cmp     esi, 64h ; 'd'
.text:00401025                 jle     short loc_40100E
.text:00401027                 push    edi
.text:00401028                 push    offset unk_407030
.text:0040102D                 call    sub_401040
.text:00401032                 add     esp, 8
.text:00401035                 xor     eax, eax
.text:00401037                 pop     edi
.text:00401038                 pop     esi
```

编译器对循环结构的优化

- 强度削弱：用等价的低周期的指令代替高周期的指令。
- 减少分支：减少跳转。

- 代码外提：编译器在编译程序时会检测循环结构中时候有重复的操作，在对循环结构中语句块的执行结构没有任何影响的情况下（外提代码必须是循环结构中不会修改的值），可选择相同的代码进行外提，以减少循环语句块中的执行代码，提高循环执行效率。