Retrieving the current EIP in C/C++

Retrieving the current EIP in C/C++

Posted: 30/01/2022

The following inline assembly sample will retrieve the current EIP value whilst preserving the call-stack in 32-bit/WoW64 applications:

```
#define GET_EIP(EIP_VALUE) _asm push eax \
_asm push ebx \
_asm lea ebx, EIP_VALUE \
_asm _emit 0xE8 \
_asm _emit 0x00 \
_asm _emit 0x00 \
_asm _emit 0x00 \
_asm _emit 0x00 \
_asm _emit 0x8B \
_asm _emit 0x04 \
_asm _emit 0x24 \
_asm _emit 0x83 \
_asm _emit 0x04 \
_asm _emit 0x24 \
_asm _emit 0x08 \
_asm _emit 0xC3 \
_asm _emit 0x83 \
_asm _emit 0xC0 \
_asm _emit 0x0F \
_asm _emit 0x89 \
_asm _emit 0x03 \
_asm _emit 0x5B \
_asm _emit 0x58
```

The opcodes must be written directly using _emit to allow relative calls to be used - this also makes it easier to count the number of bytes to jump forward.

This generates the following assembly block:

```
; preserve registers
```

```
00401012 [50] PUSH EAX
00401013 [53] PUSH EBX

; get output address

00401014 [8D9D FCFFFFFF] LEA EBX, DWORD PTR SS:[EBP - 4]

; call 0x00000000 (relative)

0040101A [E8 00000000] CALL 0x0040101F

; store original return address from stack

0040101F [8B0424] MOV EAX, DWORD PTR SS:[ESP]

; add 0x8 to the return address (to return to 0x401027 in this example)

00401022 [830424 08] ADD DWORD PTR SS:[ESP], 0x8
00401026 [C3] RETN

; add 0xF to the original return value (to store the address directly after this block -
0x40102E in this example)

00401027 [83C0 0F] ADD EAX, 0xF

; store the eip

0040102A [8903] MOV DWORD PTR DS:[EBX], EAX

; restore register values

0040102C [5B] POP EBX
0040102D [58] POP EAX
```

Example of use:

```
int main()
{
DWORD dwEIP = 0;

// break
_asm int 3
```

```
// add 'nop' instructions for reference
_asm nop
_asm nop
_asm nop

// get EIP
GET_EIP(dwEIP)

// add 'nop' instructions for reference
_asm nop
_asm nop
_asm nop

// print prevously stored EIP value
printf("dwEIP: 0x%08x\n", dwEIP);

// break
_asm int 3

return 0;
}
```

The output value is correct as shown below: