

Re: Role of the code segment register during "far" CALLs

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2006-08/msg00796.htm>

- *From:* "Skywing [MVP]" <skywing_NO_SPAM@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 18 Aug 2006 19:33:05 -0400
-

Actually, NT uses the user/supervisor bit on pages instead of segment descriptors for that. If you look at the user mode cs/ss descriptors (or any of the others except for fs on x86), you will see that they allow access for the full 4GB of address space.

```
(cc70.c864): Break instruction exception - code 80000003 (first chance)
eax=7ffda000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004
edi=00000005
eip=7c901230 esp=0093ffcc ebp=0093fff4 iopl=0  nv up ei pl zr na pe
nc
cs=001b  ss=0023  ds=0023  es=0023  fs=0038  gs=0000
efl=00000246
ntdll!DbgBreakPoint:
7c901230 cc int 3
0:001> dg @cs
P Si Gr Pr Lo
Sel Base Limit Type l ze an es ng Flags
```

```
-----
001B 00000000 ffffffff Code RE Ac 3 Bg Pg P Nl 00000cfb
0:001> dg @ss
P Si Gr Pr Lo
Sel Base Limit Type l ze an es ng Flags
```

```
-----
0023 00000000 ffffffff Data RW Ac 3 Bg Pg P Nl 00000cf3
```

If you look at some pages in the kernel debugger, you'll see that they use the user/supervisor bit (displayed as "U" or "K" by the debugger).

```
lkd> !pte nt
VA 804d7000
PDE at 00000000C0602010 PTE at 00000000C04026B8
contains 0000000004001E3 contains 0000000000000000
pfn 400 -GLDA--KWEV LARGE PAGE 4d7
```

```
lkd> !pte ntdll
VA 7c900000
PDE at 00000000C0601F20 PTE at 00000000C03E4800
```

Re: Role of the code segment register during "far" CALLs

contains 000000003E9EC067 contains 800000000BFEF025
pfn 3e9ec ---DA--UWEV pfn bfef -----A--UR-V

Windows does not use segmentation to restrict access to kernel memory.

--
Ken Johnson (Skywing)
Windows SDK MVP

"anton bassov" <soviet_bloke@xxxxxxxxxxx> wrote in message
news:1155942172.290464.175410@xx

Hi Skywing

Windows uses a flat address space and does not utilize segmentation
(other
than to allow for a convenient way to access the TEB through fs on x86 or
gs
on x64).

If Windows did not utilize segmentation, how would it separate
privileged code and stack from non-privileged ones???? Therefore, the
above statement holds only as long as you don't cross the privilege
level.

Anton Bassov

Skywing [MVP] wrote:

Windows uses a flat address space and does not utilize segmentation
(other
than to allow for a convenient way to access the TEB through fs on x86 or
gs
on x64).

--
Ken Johnson (Skywing)
Windows SDK MVP

"Ranjit" <ranjitiyer@xxxxxxxxxxx> wrote in message
news:1155937472.309032.37300@xx

Hello,

My question is in view of the memory management unit's
role in address
translation, whereby for example, to access code (CALL
<address>), the

Re: Role of the code segment register during "far" CALLs

CS register is first used as a selector to retrieve a Segment Descriptor that specifies the base of the code segment, to which an offset gets added to form a linear address which is then forwarded to the paging unit to generate a physical address. Fine?

The question is, at what step, in the course of executing a far CALL does the CS register change, and who changes it and where are segment information for each module stored (PE file?). Since a far call crosses code segment boundaries i presume that the code segment register also should change to point to the code segment of the module in which the called function resides. Just not sure when it happens and who makes it happen after the far CALL instruction is issued.

I might also be doing something wrong in my experiments, but i don't see the CS register value (register window) change when i call CoInitialize (which resides in ole32.dll) from my application's main method.

Thanks for any guidance on this one.
Ranjit