

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/msg00794.htm>

- *From:* "anton bassov" <soviet_bloke@xxxxxxxxxxx>
 - *Date:* 18 Aug 2006 16:02:52 -0700
-

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 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?

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

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