The Intel 80386, part 13: Calling conventions

devblogs.microsoft.com/oldnewthing/20190205-00

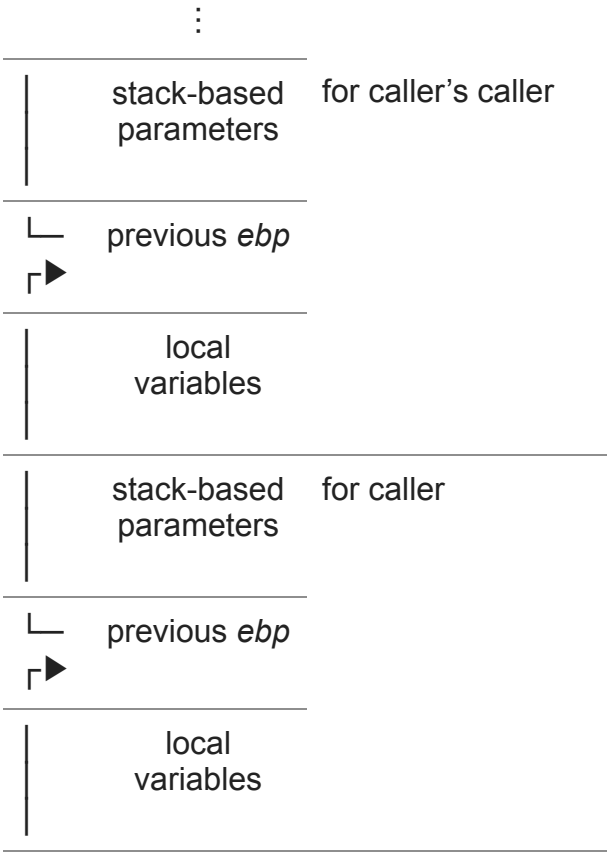
February 6, 2019

Raymond Chen

I covered calling conventions for x86 some time ago. Consider that information incorporated by reference.

Nonstandard calling conventions are permitted, provided the caller and callee agree what the convention is. Such nonstandard calling conventions may arise as the result of link-time code generation, because the linker can create a custom calling convention and simultaneously alter all the call sites to conform to it.

In order to access stack-based inbound parameters, it is conventional (but not required) to build a stack frame, using the *ebp* register as the frame pointer. The *ebp* register points to the previous stack frame, which results in a linked list of stack frames headed by the *ebp* register.



	stack-based parameters	for current function
└	previous *ebp*	← current *ebp*
	local variables	← current *esp*

Use of *ebp* as a frame pointer is not mandatory, and it was fashionable at the time not to do so in order to permit general-purpose use of an additional register. This technique is known as *frame pointer omission*, or FPO.[1] If code was compiled with FPO, then the debugger's `k` command will require additional debugging information in the PDB file in order to follow the stack trace past an FPO frame.

As of this writing, the guidance is not to use FPO. This permits Watson to generate full stack traces and allows more intelligent bucketing of crashes on the back end.

If you end up debugging a module that was compiled with FPO, but for which you do not have debugging information that includes FPO information, then stack traces will unceremoniously stop when they read an FPO function. Next time, we'll look at how to rescue those stack traces.

[1] Not to be confused with the other FPO.

Raymond Chen

Follow