More on trivial functions like CopyRect and EqualRect

devblogs.microsoft.com/oldnewthing/20200901-00

September 1, 2020

Raymond Chen

Some time ago, I discussed trivial functions like `CopyRect` and `EqualRect` and why they even exist at all.

Piotr Siódmak wondered if you could just use `memcpy` to copy a rectangle. "It's already there and probably already used by other parts of code, so it's 0 bytes of overhead."

One of the features of `CopyRect` and friends is that they all take far pointers to `RECT`. This means that you can use them to manipulate rectangles received from the operating system, even if your program uses the small or medium memory model.

The `memcpy` version of `CopyRect` would actually be `_fmemcpy` in order to support far pointers. And since it's a C runtime function, it would follow the `__cdecl` calling convention, which is caller-clean. And you would have to pass an extra parameter to represent the size of a rectangle.

```
b8 08 00        mov  ax, 8          ; sizeof(RECT)[1]
50              push ax
c4 5e f0        les  bx, [bp-10]    ; es:bx -> source rect
53              push bx
06              push es
c4 5e ec        les  bx, [bp-14]    ; es:bx -> destination rect
53              push bx
06              push es
9a xx xx xx xx  call _fmemcpy
83 c4 0a        add  sp, 10         ; clean the stack
```

This adds seven bytes to the code size.

But it's worse than that. You also have to segment-tune your `_fmemcpy` function: Which functions are you going to put into a single segment for code swapping purposes? Maybe copying rectangles is something your program does constantly, so you want to keep it in your "hot" segment. Or maybe it's something you do only occasionally, in which case you would put it in one of the more rarely-used segments. It means that when you do get around to copying a rectangle, you may have to pause to load the `_fmemcpy` function off the floppy drive.

And since the function is linked into your program, it means that if you are running five programs, each of them will have its own copy of `_fmemcpy`, with various other functions coming along for the ride in the same segments.

Why not just use the copy that the window manager already has?

The rectangle functions are in a "fixed" segment in the window manager, meaning that it is permanently loaded and cannot get swapped out. Calling it is basically free. You will never incur a segment swap by calling `CopyRect` or `EqualRect`. No need to create five private copies of a function and swap them in and out. Just use the one copy that the system already provides. That'll help you fit your program into 256KB of memory.

Bonus chatter: The `CopyRect` function is also more efficient than a general-purpose `_fmemcpy` since it knows that it's copying exactly eight bytes. The `_fmemcpy` function needs to support arbitrary-sized memory blocks, so it will spend extra time preparing for a block copy operation for maximum speed, even though that speed gain is not realized on such small copies.

[1] Why not use the two-byte `push 8` instruction? Because this is 8086 code, and the two-byte "push sign-extended 8-bit immediate" instruction didn't exist. It was first available in the 80186 processor.

Raymond Chen

Follow