Why does MS-DOS put an int 20h at byte 0 of the COM file program segment?

devblogs.microsoft.com/oldnewthing/20200309-00

March 9, 2020

Raymond Chen

The MS-DOS `.com` file format is very simple: It just a memory dump of the 16-bit address space starting at offset `0100h`, and continuing for the size of the program.

The memory below `0100h` also had a specific format, known as the *Program Segment Prefix*. There's a lot of stuff in there, but the stuff that's interesting for today's discussion are the following:

- At offset `0000h` is an `int 20h` instruction.
- At offset `0005h` is a `jmp` instruction.
- At offset `005Ch` is a file control block that contains the first command line argument, parsed as if it were a file name.
- At offset `006Ch` is a file control block that contains the second command line argument, parsed as if it were a file name.
- At offset `0080h` is the command line.

The `int 20h` is the "exit program" system call. One theory is that it is placed at offset `0000h` so that if execution runs off the end of the code segment, the instruction pointer will wrap back around to zero, and then the program will terminate.

An interesting theory, but unlikely. The odds of execution running harmlessly off the end of the code segment are slim to none.

These specific bytes are significant because they line up exactly with how CP/M organized its zero page. Keeping these important addresses the same made it easier to port CP/M programs to MS-DOS.

And CP/M put the "exit program" system call at offset `0000h` because it started each program with `0000h` on the stack. If the program executed a `ret` instruction, it would return back to zero, and exit the program. Just like if you do a `return` from `main`.

And although `int 21h` was the primary system call for MS-DOS, it supported the CP/M system call address: `call 0005h`. To further ease the porting effort from CP/M to MS-DOS, MS-DOS chose system call function codes to match the CP/M function codes.

In other words, the `int 20h` is at offset `0000h` for backward compatibility with CP/M.

Bonus chatter: The CP/M history also calls out how unlikely it is for execution to run off the end of the segment and wrap around. In order for that to happen, it would have to somehow execute through the operating system itself, because CP/M put the operating system at the highest available address. (Also, the highest available address may not be `0xFFFF` because the system could very well have less than 64KB of memory.)

Follow-up: Commenter Jim Nelson points out that this jump instruction deserves an entire article by itself, and fortunately he also provided a link to that article. It's a wild tale of deception, lies, and the A20 line.

Raymond Chen

Follow