Why does the GetVersion function report the major version in the low-order byte, and the minor version in the high-order byte?

devblogs.microsoft.com/oldnewthing/20190627-00

June 27, 2019

Raymond Chen

Laura Butler wonders whose idea it was to have the `GetVersion` function report the major version in the low-order byte, and the minor version in the high-order byte. This is completely messed up: You clearly should put the major version in the high-order byte and the minor version in the low-order byte, so that you can do things like

```
if (HypotheticalBetterGetVersion() >= 0x030A) {
  // version is at least 3.10
}
```

Instead, as things stand today, the major version in the low-order byte, and the minor version in the high-order byte, so version 3.10 is reported as the value `0x0A03`, which leads to mistakes like this:

```
// Code in italics is wrong
if (GetVersion() >= 0x0A03) {
  // incorrect check for version ≥ 3.10
}
```

Why is the version number reported in such a strange way?

Rewind to MS-DOS.

MS-DOS has a *Get Version* system call, and it returns the operating system version in the same reversed way, with the major version in the low-order byte, and the minor version in the high-order byte.

The thinking was that programs will almost always be checking the major version exclusively, because the only time you'd need to check the minor version is if you needed to check for some feature that was added in a minor version. But by definition, minor versions don't add many features (right?), so checking for the minor version should be very rare.

During this era, programs were written in assembly language, and that's the detail that shines light on the unusual format of the version number.

On the 8086, the 16-bit `AX` register can be treated as a 16-bit value, or it can be treated as two 8-bit values. If you treat it as two 8-bit values, then the high-order byte is called `AH`, and the low-order byte is called `AL`.

The 8086 instructions to compare the values in these registers are encoded as follows:

Instruction	Encoding
`CMP AL, 03h`	`3C 03`
`CMP AH, 03h`	`80 FC 03`

Since almost everyone will be comparing the major version, we should put the major version in the location that is most efficient to consume, and that would be the `AL` register. Putting the value there permits a shorter instruction encoding than if the value had been in the `AH` register.

It saves a byte!

The IBM PC came in two memory configurations, 16KB and 64KB. If you had shelled out the big bucks for the 64KB version, that one byte is 0.0015% of your total memory. Scaled to a modern 8GB system, choosing this format for the version number saves 128KB.

Windows adopted the version numbering scheme from MS-DOS, and that's why the `GetVersion` function returns the major and minor versions in reverse order.

Raymond Chen

Follow