Advantages of knowing your x86 machine code

devblogs.microsoft.com/oldnewthing/20041111-00

November 11, 2004

Raymond Chen

Next time you find yourself debugging in assembly language (which for some of us is the **only** way we debug), here are some machine code tricks you may wish to try out:

90
This is the single-byte NOP opcode. If you want to patch out code and don't want to think about it, just whack some 90's over it. To undo it, you have to patch the original code bytes back in, of course.

CC
This is the single-byte INT 3 opcode, which breaks into the debugger.

74/75
These are the opcodes for JZ and JNZ. If you want to reverse the sense of a test, you can swap one for the other. Other useful pairs are 72/73 (JB/JNB), 76/77 (JBE/JA), 7C/7D (JL/JGE), and 7E/7F (JLE/JG). You don't have to memorize any of these values; all you have to recognize is that toggling the bottom bit reverses the sense of the test. To undo this, just flip the bit a second time.

EB
This is the unconditional short jump instruction. If you want to convert a conditional jump to an unconditional one, change the 74 (say) to EB. To undo this, you have to remember what the original byte was.

00
On the other hand, if you want to convert a conditional short jump to a never-taken jump, you can patch the second byte to zero. For example, "74 1C" becomes "74 00". The jump is still there; it just jumps to the next instruction and therefore has no effect. To undo this, you have to remember the original jump offset.

B8/E8
These are the opcodes for the "MOV EAX,immed32" and the "CALL" instructions. I use them to patch out calls. If there's a call to a function that I don't like, instead of wiping it all to 90's, I just change the E8 to a B8. To undo it, change the B8 back to an E8.

It has been pointed out that this works only for functions that take zero stack parameters; otherwise, your stack gets corrupted. More generally, you can use `83 C4 XX 90 90` (ADD ESP, XX; NOP; NOP) where XX is the number of bytes you need to pop. Personally, I don't remember the machine code for these instructions so I tend to rewrite the CALL instruction so it calls the "RETD" at the end of the function.

I prefer these single-byte patches to wholesale erasure with 90 because they are easier to undo if you realize that you want to restore the code to the way it was before you messed with it.

Raymond Chen

Follow