

## Re: bits 32 oddities in NASM

---

*Source:* <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2006-03/msg01239.html>

---

- *From:* Frank Kotler <[fbkotler@xxxxxxxxxxx](mailto:fbkotler@xxxxxxxxxxx)>
  - *Date:* Thu, 23 Mar 2006 06:30:43 -0500
- 

James Daughtry wrote:

....

I was experimenting to see if I could use DOS interrupts with 32-bit registers.

Yeah, you can. It was the "bits 32" that was killin' ya, not the 32-bit registers. You can use 32-bit addressing modes in 16-bit code (but you have to keep the total offset within the 64k-1 limit). This can be *\*very\** handy.

But, since dos (and bios) interrupts are 16-bit code, Nasm needs to be generating 16-bit code to operate there. This means that using 32-bit registers will require an "override prefix byte" – 66h for an operand size, and 67h for address size. Nasm will generate these, where needed – you don't have to do it. There's a slight "penalty" in terms of "bloat" and speed, but the gains are often worthwhile.

I know I can do it with 16-bit registers, to some extent.

Yeah. You can use 16-bit registers – even addresses – in 32-bit code, too. Same override bytes. This is less often useful.

Most likely because there's some hidden emulation layer that caters to stupid people who try to write DOS in Windows. ;–)

Yes, there's an emulation layer for people who realize that not *\*everything\** requires a 32-bit cartoon interface. :)

Are interrupts real  
interrupts on Windows XP? Or do they end up calling a Win32 function?

Well, the interrupts are "real", I guess – the CPU stops what it's doing and jumps to the code pointed to by the interrupt vector table. But the code the IVT points to... may end up calling a "Windows function" – possibly at a lower level than the usual user-level API.

Re: bits 32 oddities in NASM

To be perfectly honest, I wanted to find a convenient way to do I/O without resorting to the C library or the evil Win32 API. Interrupts are simple and relatively easy to use. The C library is very simple and easy to use, for me, but I can imagine it being somewhat bloated for an assembly program. I \*hate\* the Win32 API with a well honed passion, even for simple things, and I try to avoid it. :-)

Well, if the "fake dos" emulation hides it behind your back, it's at least painless. I'd go ahead and use the dos interrupts and not worry about what goes on after that. Would you know in "real dos"?

If you really want to avoid the Win32 API, Linux or BSD would do that...

Best,  
Frank

.