Mixing 32 bit and 16 bit code with nasm

Asked 16 years, 4 months ago Modified 7 years, 5 months ago

Viewed 4k times

7

This is a low-level systems question.

I need to mix 32 bit and 16 bit code because I'm trying to return to real-mode from protected mode. As a bit of background information, my code is doing this just after GRUB boots so I don't have any pesky operating system to tell me what I can and can't do.

Anyway, I use [BITS 32] and [BITS 16] with my assembly to tell nasm which types of operations it should use, but when I test my code use bochs it looks like the for some operations bochs isn't executing the code that I wrote. It looks like the assembler is sticking in extras `0x66` and `0x67`'s which confuses bochs.

So, how do I get nasm to successfully assemble code where I mix 32 bit and 16 bit code in the same file? Is there some kind of trick?

operating-system nasm osdev assembly

Share edited Jul 10, 2017 at 19:03

Improve this question

Follow

GRUNT
32.8k ● 15 ● 121 ● 114

asked Aug 20, 2008 at 16:30

terry

4 Answers

Sorted by: Highest score (default)

7

The problem turned out to be that I wasn't setting up my descriptor tables correctly. I had one bit flipped wrong so instead of going to 16-bit mode I was going to 32-bit mode (with segments that happened to have a limit of one meg).

Thanks for the suggestions!

Terry

Share Improve this answer

Follow

answered Aug 21, 2008 at 14:43

Terry
621 ● 3 ● 10 ● 17

5

The 0x66 and 0x67 are opcodes that are used to indicate that the following opcode should be interpreted as a non-default bitness. More specifically, (and according to this link),

"When NASM is in BITS 16 mode, instructions which use 32-bit data are prefixed with an 0x66 byte, and those referring to 32-bit addresses have an 0x67 prefix. In BITS

32 mode, the reverse is true: 32-bit instructions require no prefixes, whereas instructions using 16-bit data need an 0x66 and those working on 16-bit addresses need an 0x67."

This suggests that it's bochs that at fault.

Share Improve this answer Follow

answered Aug 20, 2008 at 16:54

Curt Hagenlocher
20.9k ● 8 ● 62 ● 50

1

If you're in real mode your default size is implicitly 16 bits, so you should use BITS 16 mode. This way if you need a 32-bit operand size you add the 0x66 prefix, and for a 32-bit address size you add the 0x67 prefix.

Look at the Intel IA-32 Software Developer's Guide, Volume 3, Chapter 16 (MIXING 16-BIT AND 32-BIT CODE; the chapter number might change according to the edition of the book):

> Real-address mode, virtual-8086 mode, and SMM are native 16-bit modes.

The BITS 32 directive will only confuse the assembler if you use it outside of Protected Mode or Long Mode.

Share Improve this answer Follow

answered Aug 20, 2008 at 17:30

Nathan Fellman
127k ● 104 ● 264 ● 326

If anyone is looking up the Software Developer's Guide today (2023) it's now Volume 3, Chapter 22. intel.com/content/www/us/en/developer/articles/technical/…
– Jody Bruchon Jul 1, 2023 at 19:11

0

You weren't kidding about this being low-level!

Have you checked the generated opcodes / operands to make sure that nasm is honoring your BITS directives correctly? Also check to make sure the jump targets are correct - maybe nasm is using the wrong offsets.

If it's not a bug in nasm, maybe there is a bug in bochs. I can't imagine that people switch back to 16-bit mode from 32-bit mode very often anymore.

Share Improve this answer Follow

answered Aug 20, 2008 at 16:50
Neall
27.1k ● 5 ● 50 ● 49