Is one's complement a real-world issue, or just a historical one?

Asked 16 years, 2 months ago Modified 2 years, 2 months ago

Viewed 9k times

38

Another question asked about determining odd/evenness in C, and the idiomatic (x & 1) approach was correctly flagged as broken for one's complement-based systems, which the C standard allows for.

Do systems really exist in the 'real world' outside of computer museums? I've been coding since the 1970's and I'm pretty sure I've never met such a beast.

Is anyone actually developing or testing code for such a system? And, if not, should we worry about such things or should we put them into Room 101 along with paper tape and punch cards...?

c cpu-architecture numeric history ones-complement

Share

Improve this question

Follow

edited Oct 19, 2022 at 6:32

community wiki
3 revs, 3 users 95%
Roddy

5 I know this question is old, but anyone reading it should be aware that `(x & 1U)` **is** valid (note the `U`) for determining even/oddness even on ones-complement or sign/magnitude implementations. – R.. GitHub STOP HELPING ICE Oct 15, 2010 at 5:26

@R. Are you sure?: In 32-bit 1's comp, -1 is represented as 0xFFFFFFFE, so (0xFFFFFFFE & 0x00000001) = 0x00000000 = false. – Roddy Oct 15, 2010 at 9:21

4 @Roddy: Casting a negative signed integer is required to compute (MAX_INT+1- (-value)) on all systems. For systems which use two's-complement math, the result of the computation will have the same bit representation as the original signed integer, and many compilers will thus simply reinterpret the value without generating any code to operate on it. On systems with one's-complement or signed-magnitude computation, however, the compiler would have to generate code for the typecast to ensure defined behavior. – supercat Oct 13, 2011 at 17:40

@Supercat - Interesting, thanks! re: (MAX_INT+1- (-value)) - do you have a source for that? Is it only in C99 standard? – Roddy Oct 13, 2011 at 21:20

2 Wikipedia's article on signed number representations mentions that "the Unisys ClearPath Dorado series mainframes use ones' complement". – user2467198 Sep 8, 2014 at 21:41

9 Answers

Sorted by: Highest score (default)

I work in the telemetry field and we have some of our customers have old analog-to-digital converters that still

15

use 1's complement. I just had to write code the other day to convert from 1's complement to 2's complement in order to compensate.

So yes, it's still out there (but you're not going to run into it very often).

Share Improve this answer

Follow

answered Oct 2, 2008 at 11:47

dongola7

269 ● 1 ● 5

10 Signed-magnitude, offset-binary, and one's-complement exist for I/O, but in practice are almost always converted by code to either offset-binary or two's-complement as soon as they are read in. – supercat Oct 13, 2011 at 17:55

Could someone with the relevant experience please give some brand names, URLs, or search keywords that might help to answer the OP's question? This answer helps change my search keywords from "modern ones' complement platform" to "modern ones' complement a2d converter" but I'm sure it's possible to be more explicit. – Quuxplusone Jan 15, 2018 at 19:04

3 Am I correct that this answer is not talking about a C/C++ platform where `int` natively uses ones'-complement representation in memory, but is merely talking about a *wire format* (for talking to some peripheral device) which specifies that negative numbers in some/all fields must be encoded in ones'-complement? If so, this is similar in spirit to the RFC 791 example above. – Quuxplusone Jan 15, 2018 at 19:07

@Quuxplusone I think this is different from the RFC 971 example. The 1's complement number from an ADC would represent a digitized bipolar (positive and negative) signal. This ADC could then be used as an input directly into a 1's complement machine, which in turn could be processed in

C/C++; but I'm not aware of any such examples. There is some mention that major vendors like Analog Devices once made these types of ADCs, but even their reference materials suggest those are rare (cf. analog.com/media/en/training-seminars/design-handbooks/…) – chrstphrchvz Jun 26, 2018 at 0:42

…Such an ADC would also probably be old enough that it wouldn't be using any kind of packet or serial communication interface; instead the bits of the digitized sample would appear on pins of the IC ("parallel"). – chrstphrchvz Jun 26, 2018 at 0:42

13

This all comes down to knowing your roots.
Yes, this is technically an old technique and I would probably do what other people suggested in that question and use the modulo (%) operator to determine odd or even. But understanding what a 1s complement (or 2s complement) is always a good thing to know. Whether or not you ever use them, your CPU is dealing with those things all of the time. So it can never hurt to understand the concept. Now, modern systems make it so you generally never have to worry about things like that so it has become a topic for Programming 101 courses in a way. But you have to remember that some people actually would still use this in the "real world"... for example, contrary to popular belief there are people who *still use assembly!* Not many, but until CPUs can understand raw C# and Java, *someone* is going to still have to understand this stuff.

And heck, you never know when you might find your self doing something where you actually need to perform binary math and that 1s complement could come in handy.

Share Improve this answer

Follow

edited Sep 27, 2017 at 5:54

Antti Haapala -- Слава Україні

134k ● 23 ● 289 ● 342

answered Oct 2, 2008 at 11:32

Adam Haile

31.3k ● 60 ● 195 ● 290

8 Thanks. Agree fully that you need to have learnt it, but you shouldn't worry about it - like 6-bit bytes, and how core memory works. BTW, it's "complement", not "compliment". One's compliment could be "that's a nice sign bit you're wearing today". – Roddy Oct 2, 2008 at 13:36

I do a lot of interviewing of software engineers, using Steve Yegge's Five Areas (steve.yegge.googlepages.com/…), and you better believe that one of the areas is Bits and Bytes. If you want to be a good SDE, you must understand binary number systems, and ones and twos complement is part of that. I would hope that any decent Computer Science course covers this in the Computer organisation class(es).
– Josh Glover Dec 26, 2009 at 6:45

@Roddy: 6-bit bytes are actually not allowed in C or C++: the minimum amount of bits in a byte (which C / C++ define as `sizeof(char)`) is 8. – David Stone Oct 25, 2013 at 17:43

@DavidStone actually `sizeof(char)` is defined to be `1`. What you want to refer to is `CHAR_BIT` instead. – Ruslan Jul 21, 2017 at 9:14

1 @Ruslan: Sorry, what I wrote was ambiguous. I was trying to say that C and C++ define a byte as `sizeof(char)`, and the minimum number of bits in that is 8. – David Stone Jul 21, 2017 at 13:01

10

The CDC Cyber 18 I used back in the '80 was a 1s complement machine, but that's nearly 30 years ago, and I haven't seen one since (however, that was also the last time I worked on a non-PC)

Share Improve this answer

Follow

answered Oct 9, 2008 at 8:00

James Curran

103k ● 37 ● 185 ● 262

10

RFC 791 p.14 defines the IP header checksum as:

> The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

So one's complement is still **heavily** used in the real world, in every single IP packet that is sent. :)

Share Improve this answer

Follow

answered Dec 26, 2009 at 6:41

community wiki
Josh Glover

5 Hmm, well... I was referring to one's complement as a means of representing negative integers, rather as a means of bitwise inversion, but you probably knew that. +1 for effort :-) – Roddy Dec 26, 2009 at 20:06

4 A fun fact about the ones'-complement checksum is that if one reads pairs of bytes in the source data using 16-bit words and stores the two-byte checksum likewise, the same code will work for big-endian and little-endian architectures. – supercat Jun 16, 2016 at 22:42

10

I decided to find one. The Unisys ClearPath systems have an **ANSI** C compiler (yes they call it "American National Standard C" for which even the PDF documentation was last updated in 2013. The documentation is available online;

There the signed types are all using one's complement representation, with the following properties:

Type	Bits	Range
signed char	9	-2^8+1 ... 2^8-1
signed short	18	$-2^{17}+1$ ... $2^{17}-1$
signed int	36	$-2^{35}+1$ ... $2^{35}-1$
signed long int	36	$-2^{35}+1$ ... $2^{35}-1$
signed long long int	72	$-2^{71}+1$ ... $2^{71}-1$

Remarkably, it also by default supports non-conforming `unsigned int` and `unsigned long`, which range from `0`

`... 2³⁶ - 2`, but can be changed to `0 ... 2³⁶ - 1` with a pragma.

Share Improve this answer Follow

edited Sep 27, 2017 at 5:56

community wiki
2 revs
Antti Haapala

3 FYI, "ANSI C" is a common way to refer to the first C standard (C89/C90). In other words, ANSI C == C89. – Tim Čas May 30, 2018 at 21:14

2 Out of curiosity, how does it represent `signed long long int`? Does it use a "straight" binary representation, or does it do something interesting? If the system could support a 71-bit signed value with a straight binary representation, it should have no trouble supporting a conforming 70-bit unsigned type, but it doesn't. That makes me wonder if it might use something other than a straight binary representation (which is allowable for signed types, but not unsigned ones). – supercat Aug 1, 2018 at 19:30

8

I've never encountered a one's complement system, and I've been coding as long as you have.

But I did encounter a 9's complement system -- the machine language of a HP-41c calculator. I'll admit that this can be considered obsolete, and I don't think they ever had a C compiler for those.

Share Improve this answer

Follow

answered Oct 2, 2008 at 11:30

Darron

21.6k • 5 • 51 • 54

6

We got off our last 1960's Honeyboxen sometime last year, which made it our oldest machine on site. It was two's complement. This isn't to say knowing or being aware of one's complement is a bad thing. Just, You will probably never run into one's complement issues today, no matter how much computer archeology they have you do at work.

The issues you are more likely to run into on the integer side are endian issues (I'm looking at you PDP). Also, you'll run into more "real world" (i.e. today) issues with floating point formats than you will integer formats.

Share Improve this answer

Follow

edited Oct 2, 2008 at 11:42

answered Oct 2, 2008 at 11:36

user7116

64k • 17 • 146 • 174

> I was on a site a few weeks ago that still had a few honeyboxen running! I was pretty shocked to see the machine powered up. – McPherrinM Dec 26, 2009 at 6:52

Funny thing, people asked that same question on comp.std.c *in 1993*, and *nobody could point to a one's*

5

complement machine that had been used back then.

So yes, I think we can confidently say that one's complement belongs to a dark corner of our history, practically dead, and is not a concern anymore.

Share Improve this answer Follow

answered Jul 13, 2016 at 19:47

community wiki
Yakov Galka

The article linked raises a question to which no one responds ("Are there still any ones-complement machines out there that the Standard wants to support? Which one(s)? ...") but given that the thread was on a different topic, I would not consider it an exhaustive survey. – Josiah Yoder Jul 21, 2017 at 16:22

@JosiahYoder: you can't provide evidence that something isn't used. The fact that no-one can point to a specific relevant architecture, not on that thread on comp.std.c nor in this question on SO, is the best evidence you can get. In case you don't know, comp.std.c is a mailing list used for discussing C-standard evolution, including by the people on the standard committee, and was a much more important communication channel back in those days. – Yakov Galka Jul 22, 2017 at 6:29

0

> Is one's complement a real-world issue, or just a historical one?

Yes, it still used. Its even used in modern Intel processors. From Intel® 64 and IA-32 Architectures Software Developer's Manual 2A, page 3-8:

> ***3.1.1.8 Description Section***
>
> Each instruction is then described by number of information sections. The "Description" section describes the purpose of the instructions and required operands in more detail.
>
> Summary of terms that may be used in the description section:
> * Legacy SSE: Refers to SSE, SSE2, SSE3, SSSE3, SSE4, AESNI, PCLMULQDQ and any future instruction sets referencing XMM registers and encoded without a VEX prefix.
> * VEX.vvvv. The VEX bitfield specifying a source or destination register (in 1's complement form).
> * rm_field: shorthand for the ModR/M r/m field and any REX.B
> * reg_field: shorthand for the ModR/M reg field and any REX.R

Share Improve this answer Follow

answered Aug 18, 2015 at 23:46

community wiki
jww

Does compiled code ever actually contain a VEX bitfield encoded in 1's complement? – Josiah Yoder Sep 1, 2015 at 19:28

@JosiahYoder of course, whenever a VEX-encoded instruction is used, it has some fields inverted. Similarly with EVEX prefix. I believe this was done to avoid clashes with 32-bit-mode instructions, which have been removed from native 64-bit mode, but must be still working in 32-bit mode along with all the SIMD extensions. – Ruslan Jul 21, 2017 at 9:21

1 @JosiahYoder if the target CPU supports AVX, then e.g. GCC will emit VEX-encoded instructions even if they can be encoded without VEX prefix (e.g. most of SSE,SSE2,... instructions). As the compilers become more capable of auto-vectorization, these instructions become more and more common in compilers' output. As for "I believe this was done...", I was referring to the inversion as it is, not compared to 2s complement. Anyway, it's actually nothing you'd come across while writing normal high-level applications — it's just a peculiarity of how instructions of certain class are encoded. – Ruslan Jul 21, 2017 at 17:15

1 @jww - Sorry for the downvote. I don't use them often. But although 1's complement is inversion of bits, it's also a number system for representing positive and negative numbers with both a +0 and -0 and a variety of other quirks compared to 2's complement. Although the docs use the term 1's complement here, they don't mean 1's complement in this broader sense. – Josiah Yoder Jul 21, 2017 at 19:34

2 First, please read the OP, it clearly asks about *systems* (i.e. architectures) in the context of the C language, not about *applications*. Second, CRC does not interpret the bit patterns as negative numbers at all, it's instead another algorithmic use of bitwise NOT, not one's complement as a negative

number representation, which the OP was asking about.
– Yakov Galka Jul 22, 2017 at 12:05