Ruby - Bug #8204

ObjectSpace.each_object(Bignum) can generate Bignums that are to small to be Bignums

04/02/2013 10:30 PM - Hanmac (Hans Mackowiak)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	2.1.0		
ruby -v:	ruby 2.1.0dev (2013-04-02 trunk 40068) [x86_64-linux]	**Backport:**	

Description

when i do:

p ObjectSpace.each_object(Bignum).to_a
=> [18446744073709551615, 3, 23942136215603892576075837148453332 05]
and again:
ObjectSpace.each_object(Bignum).to_a
=> [18446744073709551615, 3, 6332658822193905880076734888853480 2301, 0, 0, 0]

my question: why does the 3 and the zeros show up?

n = ObjectSpace.each_object(Bignum).min #=> 0
p n.class #Bignum
p n.zero? #=> false
p n < 0 # true

okay this means i have a zero, that looks like zero, but is not a zero and its class is Bignum

(but when i try in a new ruby session)

(2**64) < 0;
p bigZero = ObjectSpace.each_object(Bignum).min #=> 0
p bigZero.class #Bignum
p bigZero.zero? #=> true

okay, this time i get a zero that looks like a zero, and react like a zero, but its still a Bignum

so where does the zeros come from and why does they react so totaly different?

Associated revisions

Revision b30a6b8d1d194528a2c84b7e2c73d23a4d25cc42 - 04/03/2013 07:34 AM - nobu (Nobuyoshi Nakada)

bignum.c: Bignum zero comparison

- bignum.c (rb_big_eq): test as Fixnum if possible and get rid of zero length Bignum. [ruby-core:53893] [Bug #8204]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@40076 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision b30a6b8d - 04/03/2013 07:34 AM - nobu (Nobuyoshi Nakada)

bignum.c: Bignum zero comparison

- bignum.c (rb_big_eq): test as Fixnum if possible and get rid of zero length Bignum. [ruby-core:53893] [Bug #8204]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@40076 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision adf1c94ffe75cea7d6b12ac10809656c15d33079 - 04/03/2013 07:35 AM - nobu (Nobuyoshi Nakada)

bignum.c: hide intermediate Bignums

- bignum.c (rb_big_eq): hide intermediate Bignums not just freeing memory. [ruby-core:53893] [Bug #8204]

- object.c (rb_obj_hide): hide an object by clearing klass.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@40077 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision adf1c94f - 04/03/2013 07:35 AM - nobu (Nobuyoshi Nakada)

bignum.c: hide intermediate Bignums

- bignum.c (rb_big_eq): hide intermediate Bignums not just freeing memory. [ruby-core:53893] [Bug #8204]
- object.c (rb_obj_hide): hide an object by clearing klass.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@40077 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 04/03/2013 10:12 AM - hsbt (Hiroshi SHIBATA)

- Category set to core

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 2.1.0

#2 - 04/03/2013 11:23 AM - Anonymous

Note that you can obtain whichever Bignum you want using coerce. For example:

x = (1 << 100).coerce(42).first # => 42
x.class # => Bignum

I saw that used in test/*

I never bothered to investigate it, but I'm pretty sure this could lead to multiple bugs as MRI assumes that Bignum means big number in many places.

#3 - 04/03/2013 04:34 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r40076.
Hans, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

---

bignum.c: Bignum zero comparison

- bignum.c (rb_big_eq): test as Fixnum if possible and get rid of zero length Bignum. [ruby-core:53893] [Bug #8204]

#4 - 04/04/2013 02:27 PM - Hanmac (Hans Mackowiak)

okay that solvs a bit, so the generated Bignums are zero too, but for sample

ObjectSpace.each_object(Bignum).select &:zero?

still generates Bignum zeros and the 3 that was inside the Bignum list at the beginning confuses me

#5 - 04/16/2013 12:18 PM - nobu (Nobuyoshi Nakada)

```
x = (1 << 100).coerce(42).first # => 42
x.class # => Bignum
```

It's a Feature.

Numeric#coerce must return a pair of converted interoperable values.

In future version, Fixnum and Bignum might be merged.
However, these commits do not intend it.

Just fixes the comparison, and hides and frees intermediate objects earlier.