

Ruby - Feature #13507

Improve performance of some Complex methods where call Numeric#real? internally

04/25/2017 07:20 AM - watson1978 (Shizuo Fujita)

Status:Open

Priority:Normal

Assignee:

Target version:

Description

Some Complex methods call Numeric#real? at f\_real\_p() using rb\_funcall(). This patch will provide optimization in f\_real\_p() when Integer/Float is given as internal objects.

Before

Calculating -----  
Complex#+5.226M (± 3.9%) i/s - 26.145M in 5.011147s  
Complex#-5.274M (± 4.4%) i/s - 26.321M in 5.001267s  
Complex#\*3.217M (± 4.7%) i/s - 16.092M in 5.013429s

After

Calculating -----  
Complex#+6.925M (± 5.4%) i/s - 34.559M in 5.006583s  
Complex#-7.124M (± 4.8%) i/s - 35.652M in 5.017364s  
Complex#\*3.880M (± 4.1%) i/s - 19.363M in 5.000170s

Test code

require 'benchmark/ips'  
  
Benchmark.ips do |x|  
 c1 = Complex(2, 3)  
 c2 = Complex(2, 3)  
  
 x.report "Complex#+" do |t|  
 t.times { c1 + c2 }  
 end  
  
 x.report "Complex#-" do |t|  
 t.times { c1 - c2 }  
 end  
  
 x.report "Complex#\*" do |t|  
 t.times { c1 \* c2 }  
 end  
  
end

Patch

<https://github.com/ruby/ruby/pull/1598>

History

#1 - 04/28/2017 05:16 AM - jzakiya (Jabari Zakiya)

Could your patch

```
-funl(real_p)
+
+inline static VALUE
+f_real_p(VALUE x)
+{
```

07/30/2025

1/2

```
+    if (RB_INTEGER_TYPE_P(x)) {
+        return Qtrue;
+    }
+    else if (RB_FLOAT_TYPE_P(x)) {
+        return Qtrue;
+    }
+    return rb_funcall(x, id_real_p, 0);
+}
```

be simplified to the following?

```
-funl(real_p)
+
+inline static VALUE
+f_real_p(VALUE x)
+{
+    if (RB_INTEGER_TYPE_P(x) || RB_FLOAT_TYPE_P(x)) {
+        return Qtrue;
+    }
+    return rb_funcall(x, id_real_p, 0);
+}
```

## #2 - 06/24/2019 08:21 PM - jeremyevans0 (Jeremy Evans)

- Tracker changed from Bug to Feature

- Backport deleted (2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN)