Ruby trunk - Bug #15779

After NoMemoryError, ruby freezes and takes 100% CPU

04/20/2019 11:57 AM - buzztaiki (Taiki Sugawara)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.2p47 (2019-03-13 revision 67232) [x86_64-linux]	**Backport:**	2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN

Description

Run following reproduce code, ruby freezes and takes 100% CPU.

```
require 'open-uri'

begin
  "a" * 10000000000
ensure
  p open('https://www.ruby-lang.org/')
end
```

But interestingly, the following code does not reproduce this issue.

```
require 'open-uri'

begin
  begin
    "a" * 10000000000
  rescue NoMemoryError
    raise
  end
ensure
  p open('https://www.ruby-lang.org/')
end
```

It was also reproduced when put sleep 100 in ensure clause, run it and hit Ctrl-C. But puts 'XXX' does not reproduce it.

History

#1 - 04/20/2019 12:15 PM - buzztaiki (Taiki Sugawara)

Sorry, I wote bug report in Japanese to ruby-core.

In English:

Subject: After NoMemoryError, ruby freezes and takes 100% CPU

Description:
Run following reproduce code, ruby freezes and takes 100% CPU.

```
require 'open-uri'

begin
  "a" * 10000000000
ensure
  p open('https://www.ruby-lang.org/')
end
```

But interestingly, the following code does not reproduce this issue.

```
require 'open-uri'

begin
  begin
    "a" * 10000000000
  rescue NoMemoryError
```

```
    raise
  end
ensure
  p open('https://www.ruby-lang.org/')
end
```

It was also reproduced when put sleep 100 in ensure clause. But puts 'XXX' does not reproduce it.

#2 - 04/20/2019 12:35 PM - buzztaiki (Taiki Sugawara)

- Description updated

- Subject changed from NoMemoryError □□ ensure□ ruby□ CPU 100% □□□□□□ to After NoMemoryError, ruby freezes and takes 100% CPU

#3 - 04/20/2019 12:40 PM - buzztaiki (Taiki Sugawara)

I rewrote this issue to English.

#4 - 04/29/2019 01:36 AM - buzztaiki (Taiki Sugawara)

- File strace.txt added

In this situation, I want ruby to finish GC quickly or to crash.
In my production code, this issue has been avoided by not creating huge objects and by using rescue and re-raise.

I attach strace log (strace -ttt -o strace.txt ruby a.rb).
It seems to cause loop of Resource temporarily unavailable.

```
...
1556496315.537693 read(6, 0x55794d88b333, 5) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538029 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538245 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538404 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538524 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538612 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538682 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538751 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538820 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538889 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.538958 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539027 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539094 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539161 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539320 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539401 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
1556496315.539470 read(4, 0x7ffeaf4b5f40, 8) = -1 EAGAIN (Resource temporarily unavailable)
...
```

#5 - 04/29/2019 01:37 AM - buzztaiki (Taiki Sugawara)

I test this reproduction code at ruby 2.5.5 (ruby 2.5.5p157 (2019-03-15 revision 67260) [x86_64-linux]). In this version, this issue is not reproduced (ruby is not frozen).

```
~/opt/ruby/2.5.5/bin/ruby a.rb
#<StringIO:0x0000560b317f53b8 @base_uri=#<URI::HTTPS https://www.ruby-lang.org/>, @meta={"server"=>"Cowboy", "
strict-transport-security"=>"max-age=31536000", "content-type"=>"text/html", "etag"=>"W/\"91e880089832f50e7e7b
f1371600cc60\"", "x-frame-options"=>"SAMEORIGIN", "via"=>"1.1 vegur, 1.1 varnish", "content-length"=>"424", "a
ccept-ranges"=>"bytes", "date"=>"Mon, 29 Apr 2019 01:11:21 GMT", "age"=>"1800", "connection"=>"keep-alive", "x
-served-by"=>"cache-hnd18749-HND", "x-cache"=>"HIT", "x-cache-hits"=>"1", "x-timer"=>"S1556500281.267699,VS0,V
E6", "vary"=>"Accept-Encoding"}, @metas={"server"=>["Cowboy"], "strict-transport-security"=>["max-age=31536000
"], "content-type"=>["text/html"], "etag"=>["W/\"91e880089832f50e7e7bf1371600cc60\""], "x-frame-options"=>["SA
MEORIGIN"], "via"=>["1.1 vegur", "1.1 varnish"], "content-length"=>["424"], "accept-ranges"=>["bytes"], "date"
=>["Mon, 29 Apr 2019 01:11:21 GMT"], "age"=>["1800"], "connection"=>["keep-alive"], "x-served-by"=>["cache-hnd
18749-HND"], "x-cache"=>["HIT"], "x-cache-hits"=>["1"], "x-timer"=>["S1556500281.267699,VS0,VE6"], "vary"=>["A
ccept-Encoding"]}, @status=["200", "OK"]>
Traceback (most recent call last):
a.rb: failed to allocate memory (NoMemoryError)
```

#6 - 04/30/2019 03:52 AM - wanabe (_ wanabe)

I think it is from r58380.

before r58380, rb_memerror calls rb_longjmp via rb_exc_raise and clear raised_flag.
But now rb_memerror keeps raised_flag and just calls EC_JUMP_TAG.

rb_threadptr_execute_interrupts should raise Interrupt exception on hitting ctrl-c,
but just return immediately because th->raised_flag is truthy.

I've confirmed above behavior with the script to use miniruby.

```
begin
  "a" * 10000000000
ensure
  Thread.new(Thread.current) do |t|
    Thread.pass
    t.raise Interrupt
    sleep 0.01
    STDERR.puts "shutting down..."
    Process.kill :KILL, $$
  end
  sleep
end
```

#7 - 04/30/2019 06:32 AM - wanabe (_ wanabe)

How about this?

```
diff --git a/vm.c b/vm.c
index 41064f07c3..be394b0dae 100644
--- a/vm.c
+++ b/vm.c
@@ -1895,7 +1895,7 @@ vm_exec(rb_execution_context_t *ec, int mjit_enable_p)
     }
     else {
        result = ec->errinfo;
-        rb_ec_raised_reset(ec, RAISED_STACKOVERFLOW);
+        rb_ec_raised_reset(ec, RAISED_STACKOVERFLOW | RAISED_NOMEMORY);
         while ((result = vm_exec_handle_exception(ec, state, result, &initial)) == Qundef) {
             /* caught a jump, exec the handler */
             result = vm_exec_core(ec, initial);
```

#8 - 05/16/2019 05:07 PM - buzztaiki (Taiki Sugawara)

Thank you for your reply!
In my environment, this patch with ruby 2.6.3 has been fixed this issue (ruby was not frozen)!!

```
~/opt/ruby/bug15779_v2_6_3_fix/bin/ruby ~/tmp/a.rb
#<StringIO:0x000055914e06fda8 @base_uri=#<URI::HTTPS https://www.ruby-lang.org/>, @meta={"server"=>"Cowboy", "
strict-transport-security"=>"max-age=31536000", "content-type"=>"text/html", "etag"=>"W/\"91e88008983
2f50e7e7bf1371600cc60\"", "x-frame-options"=>"SAMEORIGIN", "via"=>"1.1 vegur, 1.1 varnish", "content-length"=>
"424", "accept-ranges"=>"bytes", "date"=>"Thu, 16 May 2019 17:04:03 GMT", "age"=>"1082", "connection"=>"keep-a
live", "x-served-by"=>"cache-hnd18751-HND", "x-cache"=>"HIT", "x-cache-hits"=>"1", "x-timer"=>"S1558026243.322
961,VS0,VE0", "vary"=>"Accept-Encoding"}, @metas={"server"=>["Cowboy"], "strict-transport-security"=>["max-age
=31536000"], "content-type"=>["text/html"], "etag"=>["W/\"91e880089832f50e7e7bf1371600cc60\""], "x-frame-optio
ns"=>["SAMEORIGIN"], "via"=>["1.1 vegur", "1.1 varnish"], "content-length"=>["424"], "accept-ranges"=>["bytes"
], "date"=>["Thu, 16 May 2019 17:04:03 GMT"], "age"=>["1082"], "connection"=>["keep-alive"], "x-served-by"=>["
cache-hnd18751-HND"], "x-cache"=>["HIT"], "x-cache-hits"=>["1"], "x-timer"=>["S1558026243.322961,VS0,VE0"], "v
ary"=>["Accept-Encoding"]}, @status=["200", "OK"]>
Traceback (most recent call last):
/home/taiki/tmp/a.rb: failed to allocate memory (NoMemoryError)

cat ~/tmp/a.rb
require 'open-uri'

begin
  "a" * 10000000000
ensure
  p open('https://www.ruby-lang.org/')
end
```

Files

strace.txt	377 KB	04/29/2019	buzztaiki (Taiki Sugawara)