Ruby trunk - Bug #13930

Exception is caught in rescue above ensure

09/22/2017 02:31 PM - msauter (Michael Sauter)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.4.1p111 (2017-03-22 revision 58053) [x86_64-linux]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

Given the following code:

```
def foo
  i = 0
  3.times do |n|
    begin
      puts "a"
      i += 1
      next if i > 1
      puts "b"
    rescue => e
      puts "rescue in foo, caught #{e}"
    ensure
      puts "ensure in foo, yield from foo: #{n}"
      yield n
    end
  end
end

begin
  x = 0
  foo do |o|
    puts o
    x += 1
    raise "test" if x > 1
    puts "done yielding"
  end
rescue => e
  puts "rescue outside, caught #{e}"
ensure
  puts "ensure outside"
end
```

The output is:

```
a
b
ensure in foo, yield from foo: 0
0
done yielding
a
ensure in foo, yield from foo: 1
1
rescue in foo, caught test
ensure in foo, yield from foo: 1
1
rescue outside, caught test
ensure outside
```

So the exception raised within the yielded block is caught in the rescue block above the ensure block which yielded. That sounds wrong to me. Or is it intended? The issue seems to be with the usage of next. Also, exception is caught inside AND outside as it seems that the ensure block ends up being called twice ?!

If I change the code to this:

```
def foo
  i = 0
  3.times do |n|
    begin
      puts "a"
      i += 1
      # next if i > 1
      puts "b"
    rescue => e
      puts "rescue in foo, caught #{e}"
    ensure
      puts "ensure in foo, yield from foo: #{n}"
      yield n
    end
  end
end

begin
  x = 0
  foo do |o|
    puts o
    x += 1
    raise "test" if x > 1
    puts "done yielding"
  end
rescue => e
  puts "rescue outside, caught #{e}"
ensure
  puts "ensure outside"
end
```

Then the output is:

```
a
b
ensure in foo, yield from foo: 0
0
done yielding
a
b
ensure in foo, yield from foo: 1
1
rescue outside, caught test
ensure outside
```

I would have expected this output also when using next as above.

History

#1 - 09/22/2017 02:59 PM - msauter (Michael Sauter)

- Description updated

#2 - 09/22/2017 03:00 PM - msauter (Michael Sauter)

- Description updated

#3 - 09/22/2017 03:02 PM - msauter (Michael Sauter)

- Description updated

#4 - 09/25/2017 12:52 PM - shyouhei (Shyouhei Urabe)

- Description updated

#5 - 10/28/2018 09:13 AM - wanabe (_ wanabe)

- File bug13930.patch added

- File bug13930.disasm added

I think it's due to compile_next() and add_ensure_iseq().

The following is a reduced script result.

```
$ ruby -e 'iseq = RubyVM::InstructionSequence.compile("1.times do; begin; p :loop; next; rescue; p :NG_rescue;
 ensure; raise \"test\"; end; end rescue p $!"); File.write("bug13930.disasm", iseq.disasm); iseq.eval'
:loop
:NG_rescue
#<RuntimeError: test>
```

Attached "bug13930.disasm" is the disasm result.
"catch type: rescue st: 0001 ed: 0019" may be wrong and it might be a good to be "ed: 0006" or "ed: 0008".
Because "0009 putself" is the receiver of "0012 opt_send_without_block " in ensure clause.

I wrote "bug13930.patch" but this is incorrect because:

1. The patch has a performance issue because of throw instruction.
   - I haven't measured benchmark yet and I don't know how.
2. "next in while loop" pattern should be also fixed.
3. compile_break() / compile_redo() / compile_return() will have same issue because they use add_ensure_iseq().
   - But I can't fix them because of "break from proc-closure" error.

Files

bug13930.disasm	3.62 KB	10/28/2018	wanabe (_ wanabe)
bug13930.patch	492 Bytes	10/28/2018	wanabe (_ wanabe)