Eliminate useless catch tables and nops from lambdas

This patch frees catch tables on iseqs that don't use the catch tables. It also eliminates nop instructions from lambdas that don't need them.

Before this patch, lambdas have a "prelude nop" that is used for catch table entries:

```
$ ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
   catch table
   | catch type: break  st: 0000 ed: 0004 sp: 0000 cont: 0004
   | disasm: #<ISeq:block in <main>@-e:1 (1,8)-(1,22)> (catch: FALSE)
   | catch table
   | catch type: redo   st: 0001 ed: 0006 sp: 0000 cont: 0001
   | catch type: next   st: 0001 ed: 0006 sp: 0000 cont: 0006
   local table (size: 1, argc: 1 [opts: 0, rest: -1, post: 0, block: -1, kw: -1@-1, kwrest: -1])
   [ 1] x@0<Arg>
   0000 nop
   0001 putself           [Li]
   0002 getLocal_WC_0     x00
   0004 opt_send_without_block
   0006 leave
```

But since this particular lambda doesn't use the catch tables, there is no reason to keep the catch table or the nop instruction. This patch eliminates the nop instructions as well as the unused catch tables:

```
> ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
 0000 putobject_INT2FIX_1_     { 1}[Li]
 0001 send           <calldata!mid:times, argc:0>, block in <main>
 0004 leave
```

It's not huge, but this frees about 600kb of catch tables on RailsBench. Here is a histogram of the catch tables and sizes freed for RailsBench:

```
107269241-2d723080-69fe-11eb-9bf7-64f102251df7.png
```

The X axis is the catch table size, so the actually malloc'd size for 2 would be approximately 2 * sizeof(struct iseq_catch_table_entry). So if we have 5 tables of size 2, that would be about 5 * 2 * sizeof(struct iseq_catch_table_entry).

The size of iseq_catch_table_entry is 32:

```
(1ldb) p sizeof(struct iseq_catch_table_entry)
(unsigned long) $0 = 32
```

08/26/2022
The total catch tables freed in RailsBench is 18275, so this frees about 18275 * 32 bytes, or about 584kb:

```bash
> sum(freed_table_sizes$V1)
[1] 18275
> sum(freed_table_sizes$V1) * 32
[1] 584800
```

Instruction Sequence size is also reduced due to nop elimination, but I didn’t measure it.

Finally, this patch reduces nop calls on RailsBench from 6868813 (2.1%) to 2467772 (0.8%).

nop instructions on the master branch (265c002239):

```
[RUBY_INSNS_COUNTER] nop  6868813  (2.1%)
```

nop instructions with this patch applied:

```
[RUBY_INSNS_COUNTER] nop  2467772  (0.8%)
```

Pull request is here

Related issues:
- Related to Ruby master - Bug #18474: 938e027c seems to have caused a regression... Closed
- Related to Ruby master - Bug #18475: Yielding an element for Enumerator in an... Closed

Associated revisions

Revision 938e027c - 02/16/2021 10:00 PM - tenderlovemaking (Aaron Patterson)

Eliminate useless catch tables and nops from lambdas

Before this commit:

```ruby
$ ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
== catch table
| catch type: break st: 0000 ed: 0004 sp: 0000 cont: 0004
== catch table
| catch type: redo st: 0001 ed: 0006 sp: 0000 cont: 0001
| catch type: next st: 0001 ed: 0006 sp: 0000 cont: 0006
--|
| local table (size: 1, argc: 1 [opts: 0, rest: -1, post: 0, block: -1, kw: -10-1, kwrest: -1])
| [ 1] x@0<Arg>
| 0000  nop ( 1)[Bc]
| 0001 putself [Li]
| 0002 getlocal_WC_0 x@0
| 0004 opt_send_without_block <calldata!mid:puts, argc:1, FCALL|ARGS_SIMPLE>
| 0006 leave [Br]
--|
0000 putobject_INT2FIX_1_ ( 1)[Li]
0001 send <calldata!mid:times, argc:0>, block in <main>
0004 leave
```

After this commit:

```ruby
> ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
0000 putobject_INT2FIX_1_ ( 1)[Li]
0001 send <calldata!mid:times, argc:0>, block in <main>
0004 leave
```

Fixes [ruby-core:102418] [Feature #17613]

Co-authored-by: Alan Wu XrXr@users.noreply.github.com
Eliminate useless catch tables and nops from lambdas

Before this commit:

```ruby
$ ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
  | catch type: break st: 0000 ed: 0004 sp: 0000 cont: 0004
  |=== disasm: #<ISeq:block in <main>@-e:1 (1,8)-(1,22)> (catch: FALSE)
  | catch type: redo
  | catch type: next
  | local table (size: 1, argc: 1 [opts: 0, rest: -1, post: 0, block: -1, kw: -10-1, kwrest: -1])
  | [ 1] x@0<Arg>
  | 0000 nop
  | 0001 putself
  | 0002 getlocal_WC_0 x@0
  | 0004 opt_send_without_block <calldata!mid:puts, argc:1, FCALL|ARGS_SIMPLE>
  | 0006 leave

Fixes [ruby-core:102418] [Feature #17613]
Co-Authored-By: Alan Wu XrXr@users.noreply.github.com
```

After this commit:

```ruby
> ruby --dump=insn -e '1.times { |x| puts x }'
== disasm: #<ISeq:<main>@-e:1 (1,0)-(1,22)> (catch: FALSE)
  0000 putobject_INT2FIX_1_ ( 1)[Li]
  0001 send <calldata!mid:times, argc:0>, block in <main>
  0004 leave
```

Fixes [ruby-core:102418] [Feature #17613]
Co-Authored-By: Alan Wu XrXr@users.noreply.github.com