What I learned from LuaJIT

Excelsior JET

V8

Dart VM

LuaJIT

torch

«A SCIENTIFIC COMPUTING FRAMEWORK FOR LUAJIT»

~~deep internal insight~~

overview of interesting things

```
local p = { x = 1, y = 1 }
for i = 1, 100 do
  p = { x = p.x + i,
        y = p.y - i }
end
```

```
local p = { x = 1, y = 1 }
for i = 1, 100 do
   p = { x = p.x + i,
         y = p.y - i }
end
```

```
->LOOP:
  xorps xmm5, xmm5
  cvtsi2sd xmm5, ebp
  addsd xmm6, xmm5
  subsd xmm7, xmm5
  add ebp, +0x01
  cmp ebp, +0x64
  jle ->LOOP
  jmp ->4
```

« how does it do it? »

learning by reading sources

```lua
local p = { x = 1, y = 1, [1] = 1 }
for i = 1, 100 do
  p = { x = p.x + i,
        y = p.y - i,
        [1] = p[1] }
end
```

```
->LOOP:
  movsd [rsp+0x28], xmm6
  movsd [rsp+0x30], xmm7
  mov [rsp+0x24], eax
  mov edi, [0x000423d8]
  cmp edi, [0x000423dc]
  jb skip
  mov esi, 0x1
  mov edi, 0x000423b8
  call ->lj_gc_step_jit
  test eax, eax
  jnz ->4
skip:
  mov edi, [0x000424b0]
  mov esi, 0x00052948
  call ->lj_tab_dup
  mov esi, eax
  mov [rsp+0x20], esi
  mov edi, [0x000424b0]
  mov eax, [rsp+0x24]
  movsd xmm7, [rsp+0x30]
  movsd xmm5, [rsp+0x28]
  cmp dword [rax+0x1c], +0x01
  jnz ->4
  mov r15d, [rax+0x14]
  mov rbx, 0xfffffffb00053e50
  cmp rbx, [r15+0x20]
  jnz ->4
  xorps xmm6, xmm6
  cvtsi2sd xmm6, ebp
  addsd xmm5, xmm6
  movsd [rsp+0x10], xmm5
  mov ebx, [rsi+0x14]
  movsd [rbx+0x18], xmm5
  mov rdx, 0xfffffffb0004a188
  cmp rdx, [r15+0x8]
  jnz ->5
  subsd xmm7, xmm6
  movsd [rsp+0x18], xmm7
  movsd [rbx], xmm7
  cmp dword [rax+0x18], +0x01
  jbe ->6
  mov ebx, [rax+0x8]
  cmp dword [rbx+0xc], 0xfffeffff
  jnb ->6
  movsd xmm5, [rbx+0x8]
  movsd [rsp+0x8], xmm5
  mov edx, 0x000535d8
  call ->lj_tab_newkey
  mov ebx, eax
  mov eax, [rsp+0x20]
  movsd xmm7, [rsp+0x18]
  movsd xmm6, [rsp+0x10]
  movsd xmm5, [rsp+0x8]
  movsd [rbx], xmm5
  add ebp, +0x01
  cmp ebp, +0x64
  jle ->LOOP
  jmp ->7
```

« why does it **not** do it? »

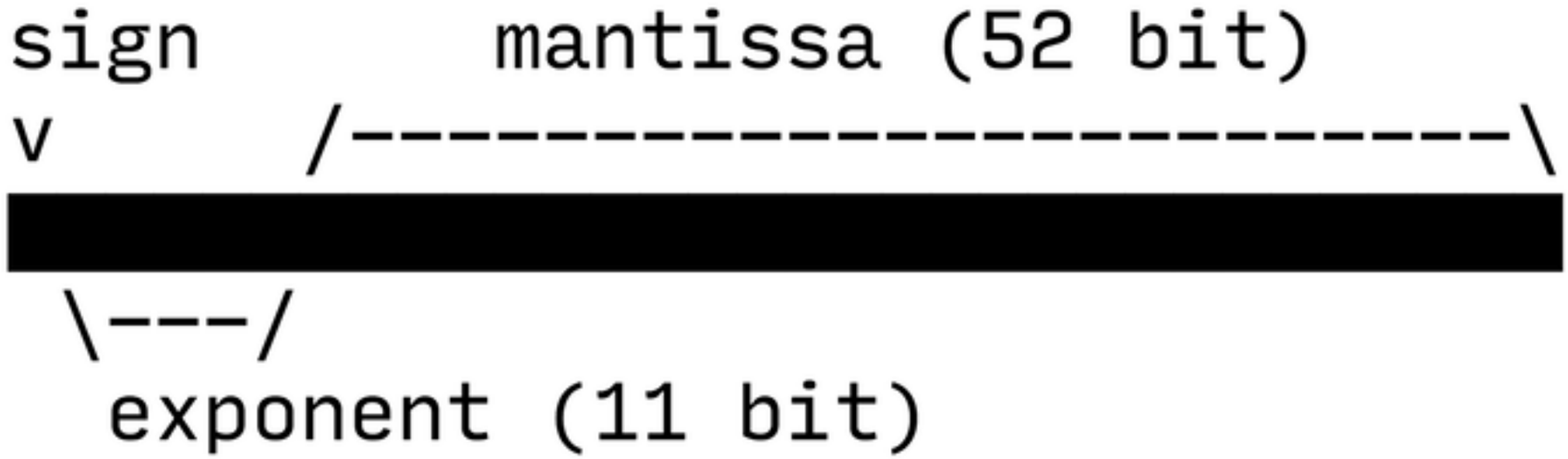
learning by fixing bugs

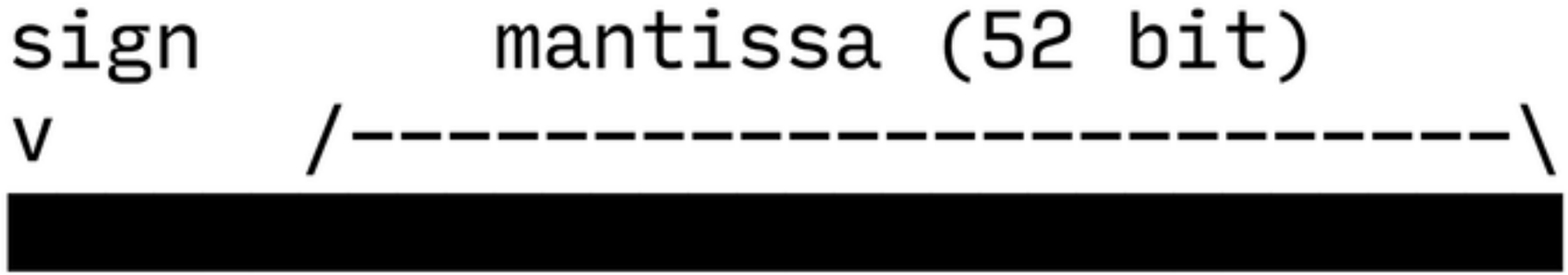
1GB memory limit

(pre v2.1)

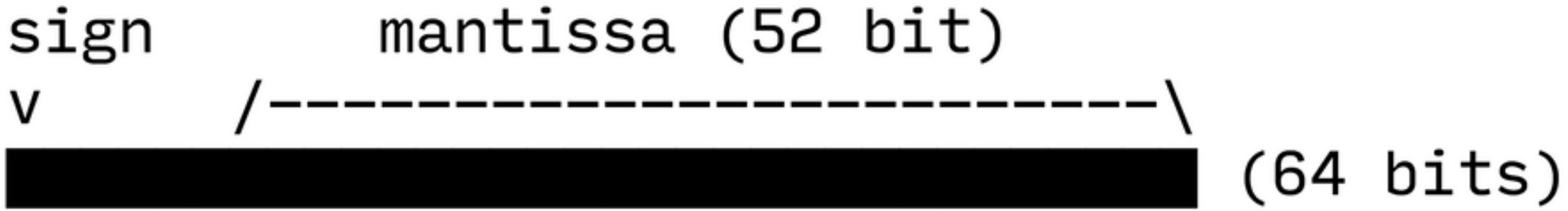
Lua is dynamically typed

NaN-tagging


sign
mantissa (52 bit)
exponent (11 bit)



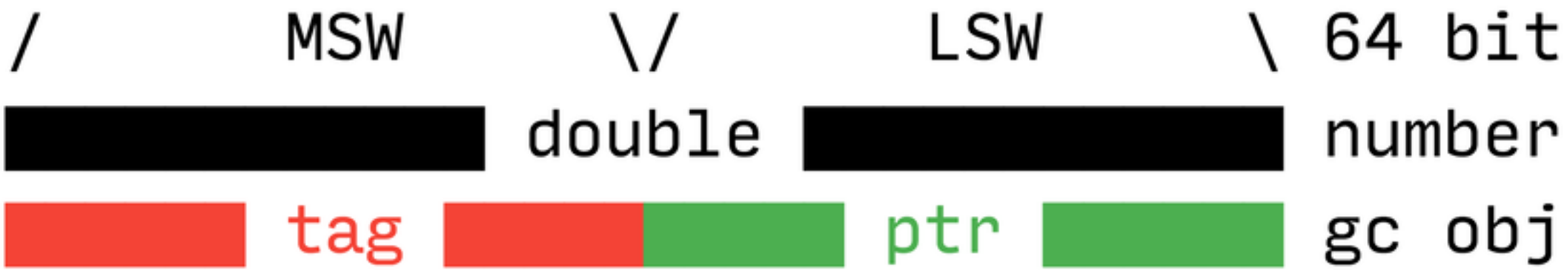
NaN: E = 7ff & M ≠ 0

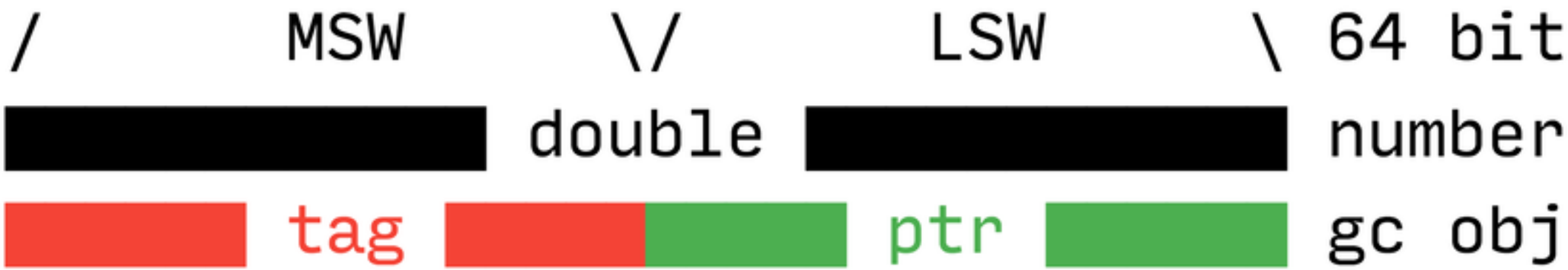


NaN: E = 7ff & M ≠ 0 (whole family of NaNs)

TValue

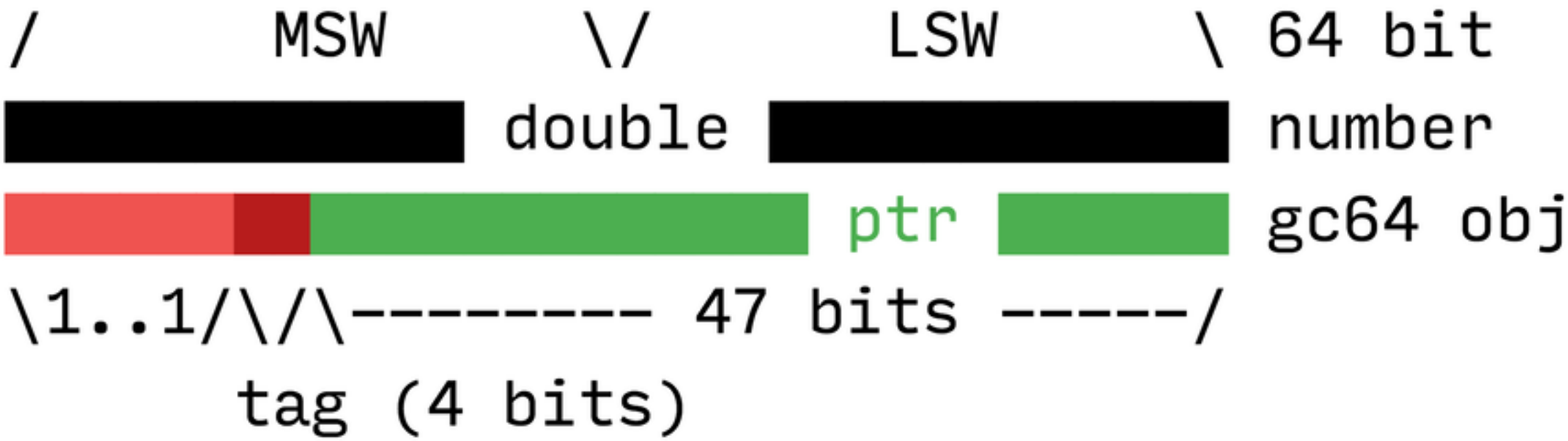
dynamically typed slot


/ MSW \/ LSW \ 64 bit
double
number
tag
ptr
gc obj



number tag < ffff0000

table tag = fffffff4 = ~11u


/ MSW \/ LSW \ 64 bit
double
number
ptr
gc64 obj
\1..1/\/\--------- 47 bits -----/
tag (4 bits)

kinda works

AArch64: 52-bit VA

changing tagging tough exercise

```
...
|
|// Macros to test operand types.
|.macro checktp, reg, tp
|  cmp dword [BASE+reg*8+4], tp
|.endmacro
|.macro checktab, reg, target
|  checktp reg, LJ_TTAB
|  jne target
|.endmacro
|
...
  case BC_TGETB:
    |  ins_ABC  // RA = dst, RB = table, RC = byte literal
    |  checktab RB, ->vmeta_tgetb
    |  mov TAB:RB, [BASE+RB*8]
...
```

DynASM

generates code that
generates code

```
case BC_TGETB:
    //|  ins_ABC  // RA = dst, RB = table, RC = byte literal
    //|  checktab RB, ->vmeta_tgetb
    //|  mov TAB:RB, [BASE+RB*8]
...
    dasm_put(Dst, 10994, LJ_TTAB, Dt6(->asize), Dt6(->array), LJ_T
```

```
| pop rax
```

```
__ pop(rax);
```

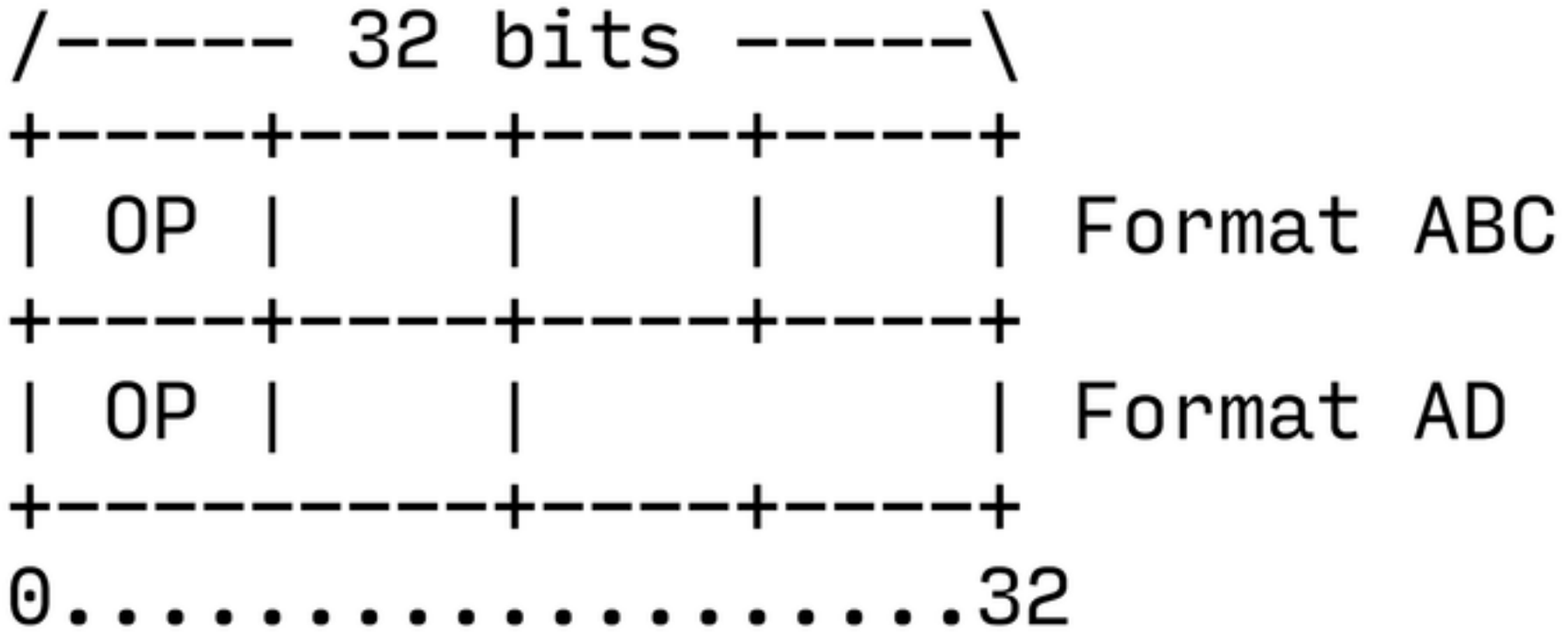
no actual understanding of types

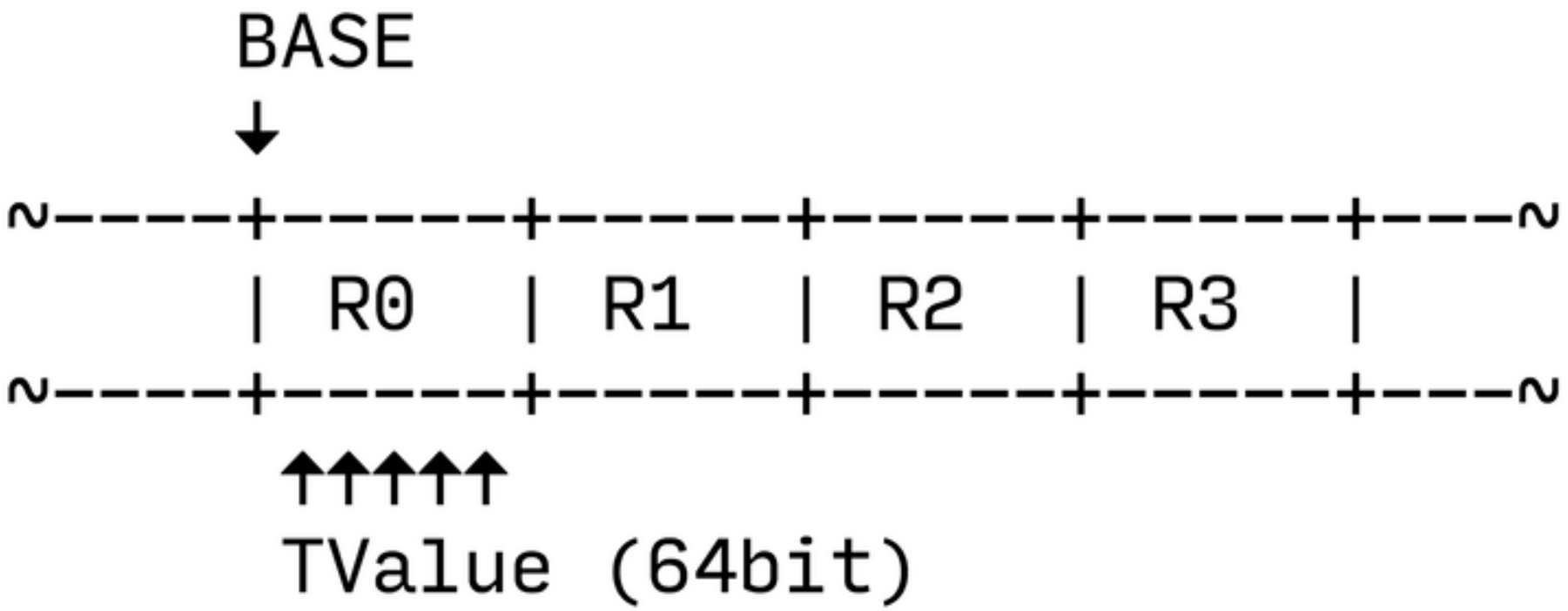
```
|   cmp dword L:RB->openupval, 0
```

```
|   cmp dword L:RB->openupval, 0
              ^^^^^^^^^^^^^^^^^ pointer
```

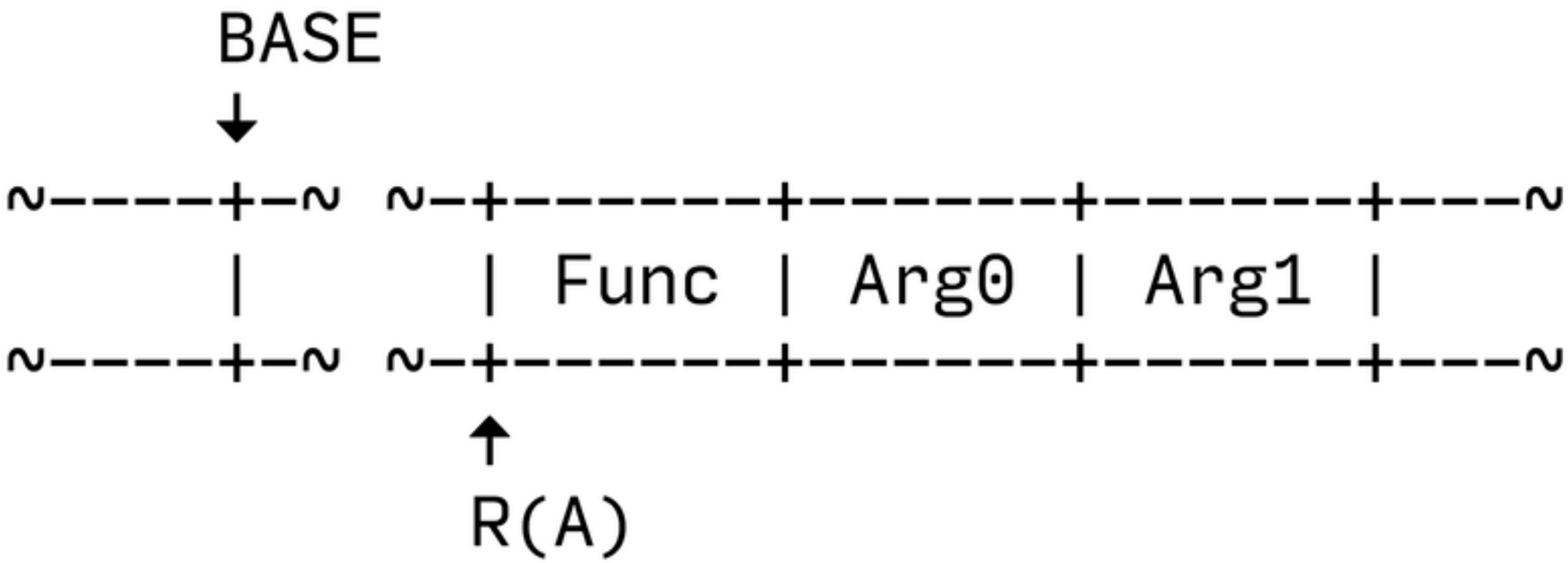
```
|   cmp aword L:RB->openupval, 0
```

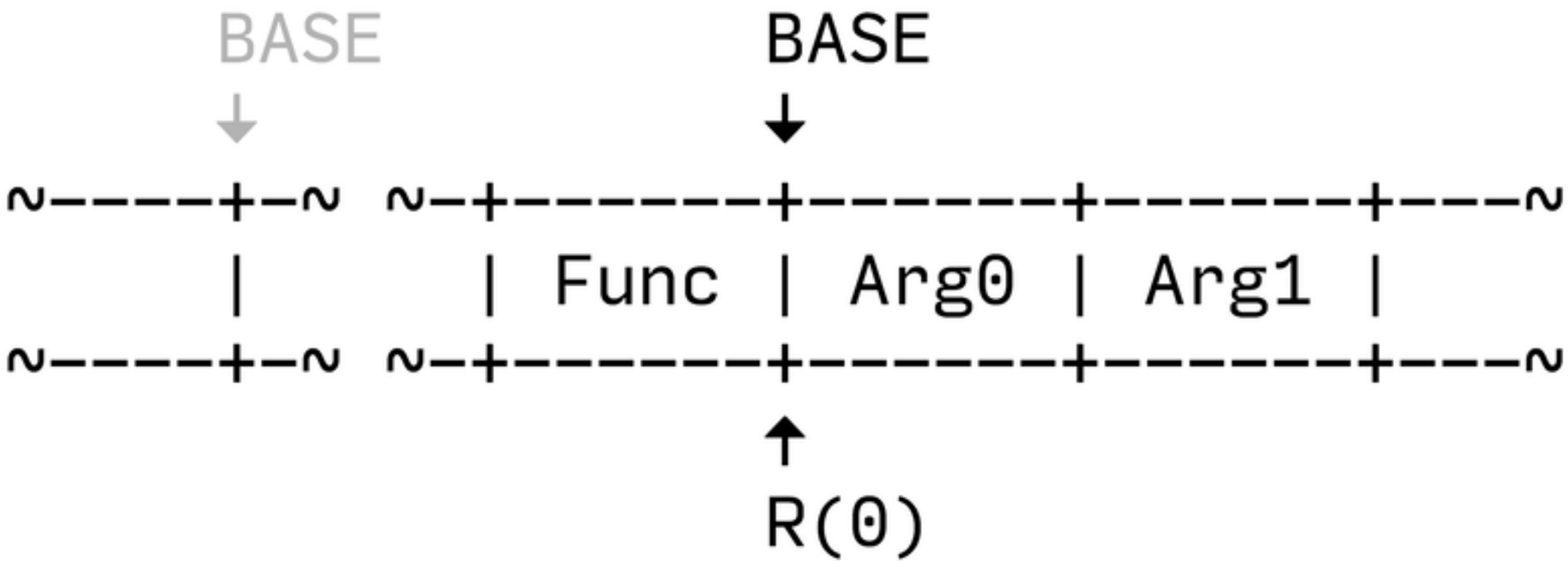
what is interpreter interpreting?


/----- 32 bits -----\
OP
OP
Format ABC
Format AD
0.....32

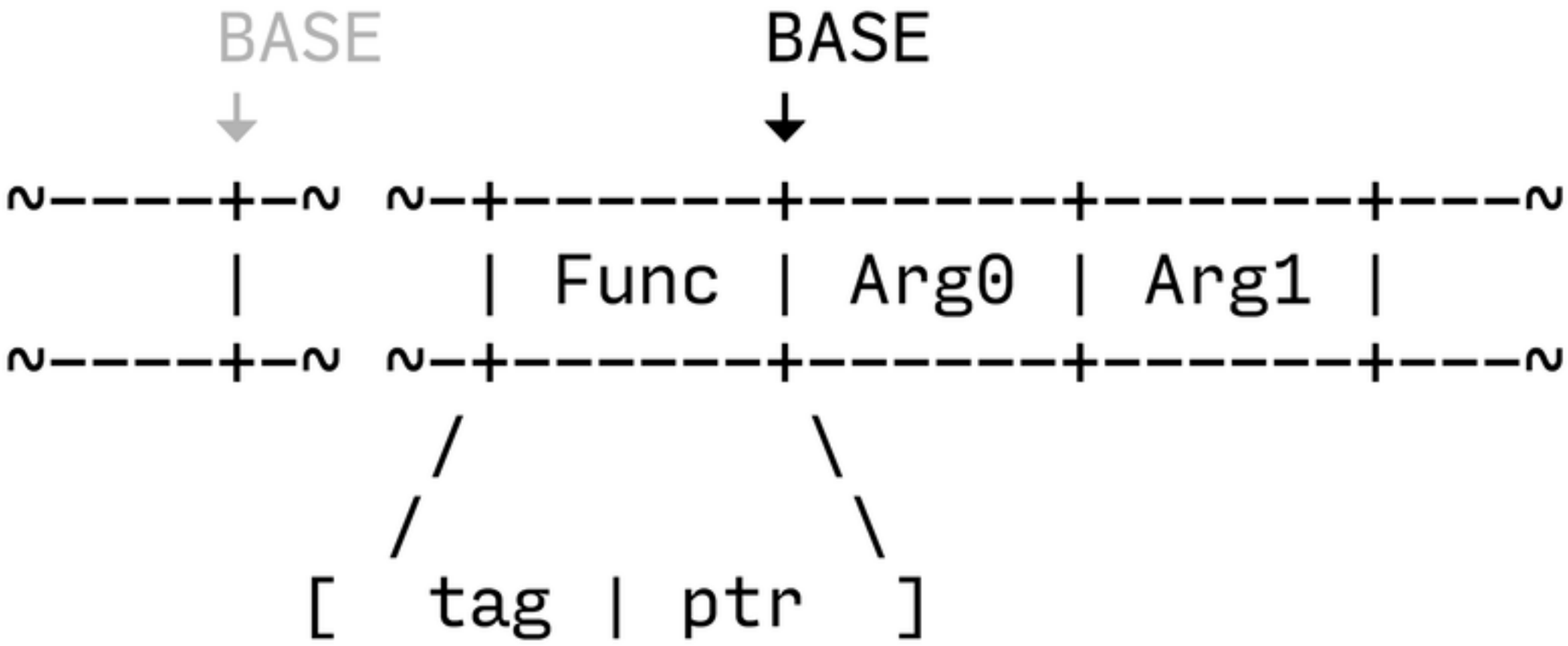

BASE
R0
R1
R2
R3
TValue (64bit)

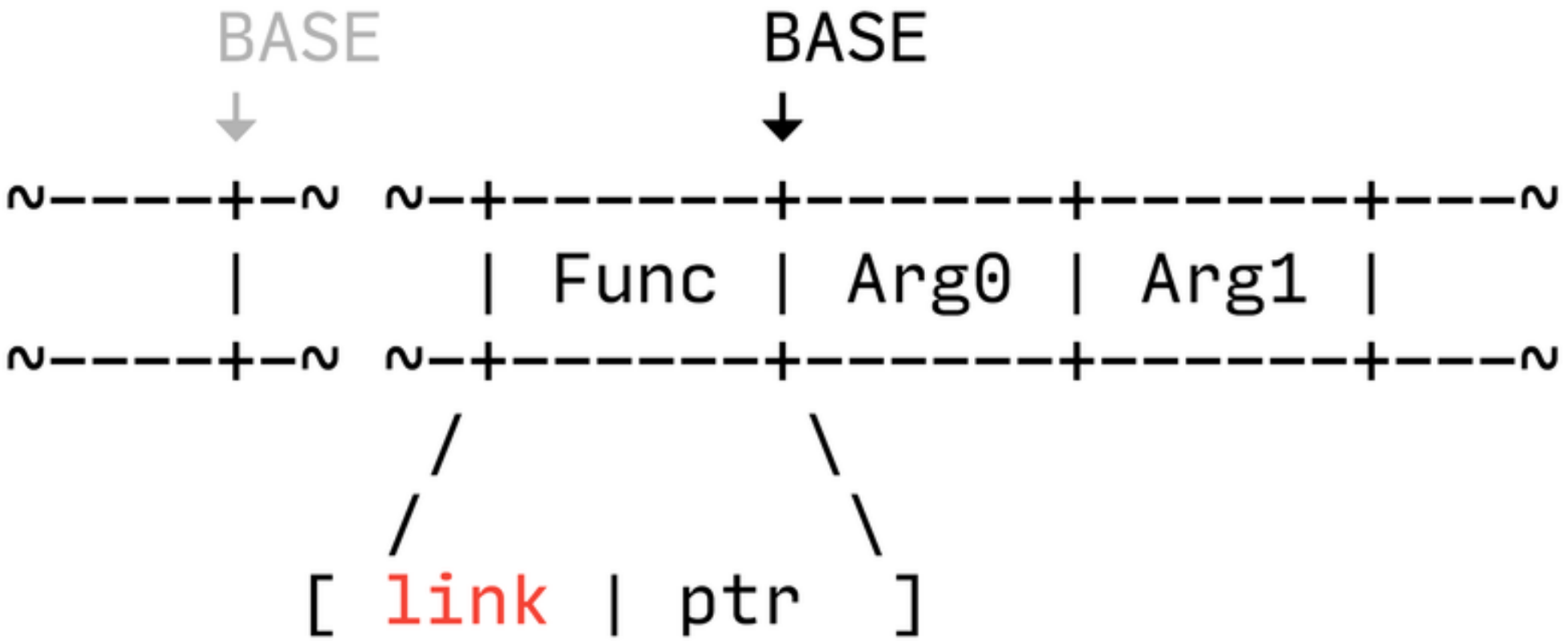
```
CALL A, ResN, ArgN
                                F <- R(A);
R(A), ..., R(A+ResN-2) <- F(R(A+1), ..., R(A+ArgN-1)), if ResN != 0
R(A), ...              <- F(R(A+1), ..., R(A+ArgN-1)), if ResN == 0
```


BASE
Func
Arg0
Arg1
R(A)


BASE
BASE
Func
Arg0
Arg1
R(0)

frame linking


BASE
BASE
Func
Arg0
Arg1
[tag | ptr]


BASE
BASE
Func
Arg0
Arg1
link
ptr

```
        link |
-------------+---------------------
      PC  00 | Lua frame
   delta 001 | C frame
   delta 010 | Continuation frame
   delta 011 | Lua vararg frame
   delta 101 | cpcall() frame
       .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

```
         link |
--------------+----------------------
      PC  00 | Lua frame
   delta 001 | C frame
   delta 010 | Continuation frame
   delta 011 | Lua vararg frame
   delta 101 | cpcall() frame
        .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

when unwinding look at PC-1 to determine caller's BASE

```
CALL A, ... => CallerBASE = BASE - A
```

```
        link |
-------------+---------------------
     PC  00 | Lua frame
  delta 001 | C frame
  delta 010 | Continuation frame
  delta 011 | Lua vararg frame
  delta 101 | cpcall() frame
      .... etc ...

PC is 4 byte aligned
delta is 8 byte aligned
```

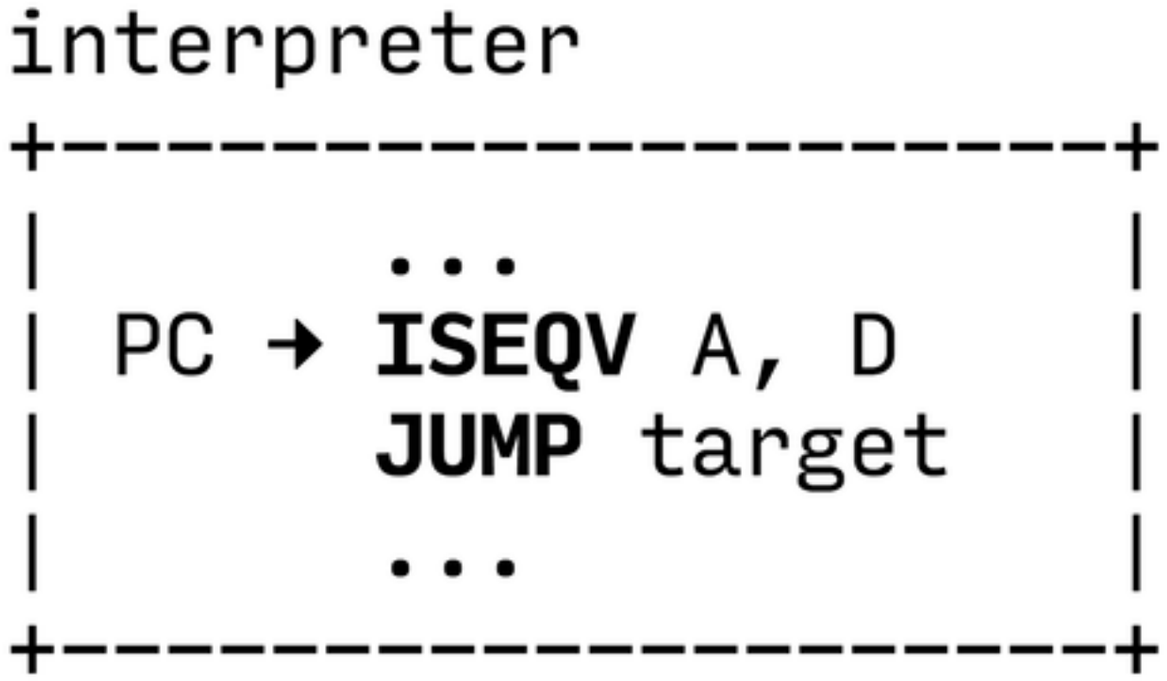
continuations specify
action to perform
when callee returns

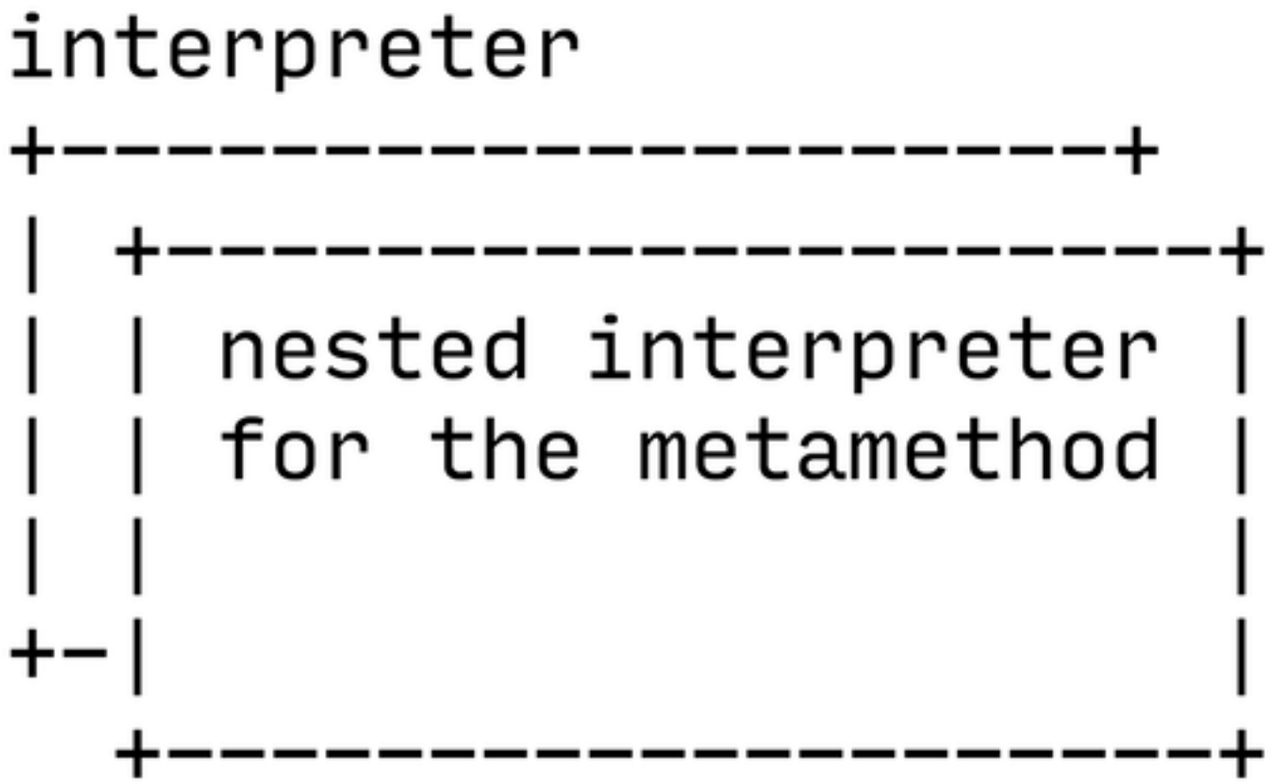
```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
```

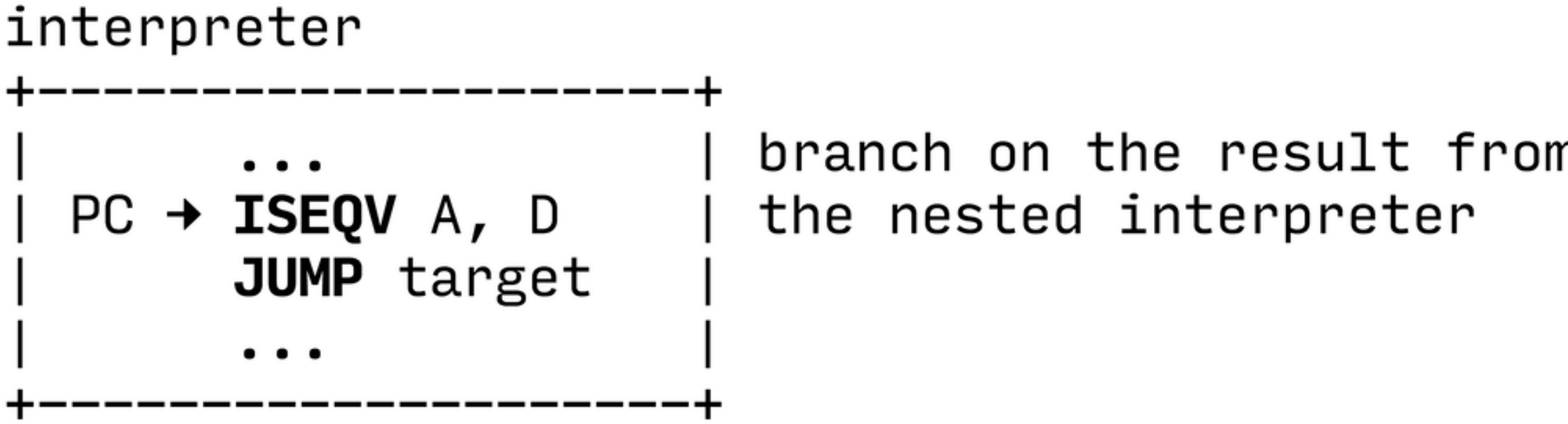
```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
```

```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
;; need to call metamethod
;; ... then branch on return
```

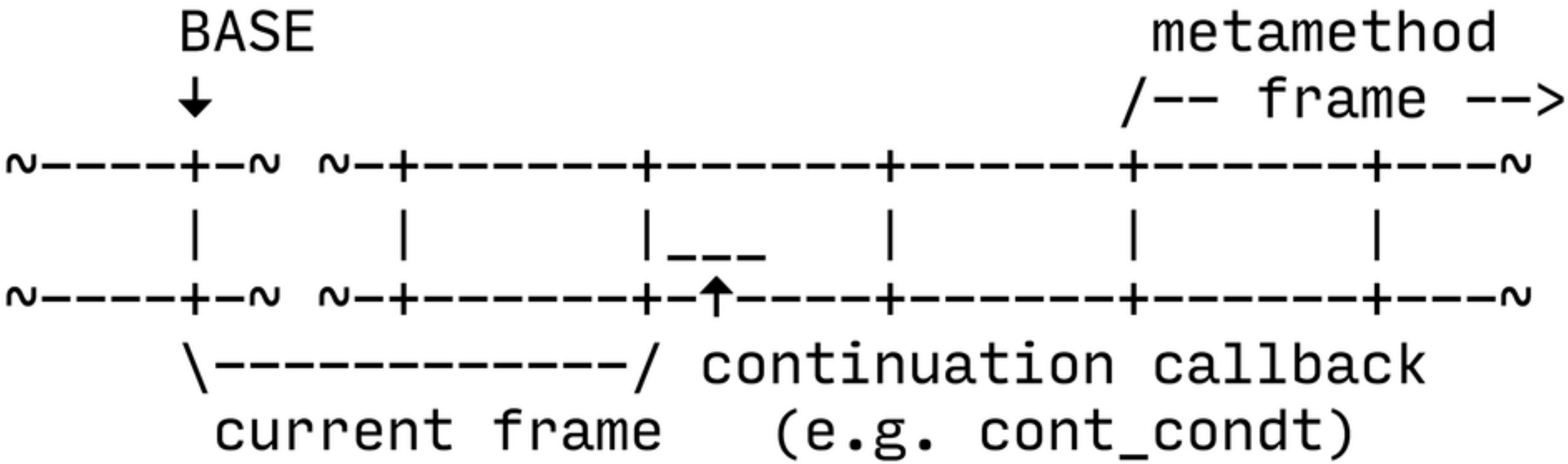
```
;; jump to target if R(A) == R(D)
ISEQV A, D
JUMP target
;; what if R(A) has __eq metamethod?
;; need to call metamethod
;; ... then branch on return
```


interpreter
...
PC → ISEQV A, D
JUMP target
...


interpreter
nested interpreter
for the metamethod


interpreter
...
PC → ISEQV A, D
JUMP target
...
branch on the result from
the nested interpreter

continuations make it simpler


BASE
metamethod
/-- frame -->
current frame
continuation callback
(e.g. cont_condt)

let's talk about DISPATCH

```
|   jmp aword [DISPATCH+OP*4]
```

```
|   jmp aword [DISPATCH+OP*4]
```

↑ can replace handlers

- hooks (~ debugging)
- profiling
- recording

```
;; hotcounting
;; loop bytecodes
FORL
ITERL
LOOP

;; function entries
FUNCF
```

```
|.macro hotloop, reg
|   mov reg, PC
|   shr reg, 1
|   and reg, HOTCOUNT_PCMASK
|   sub word [DISPATCH+reg+GG_DISP2HOT],
|            HOTCOUNT_LOOP
|   jb ->vm_hotloop
|.endmacro
```

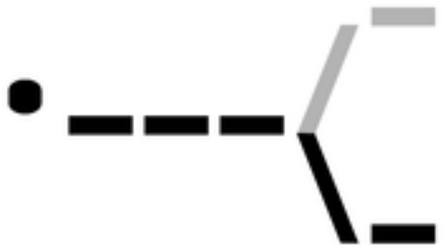
```
hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]
```

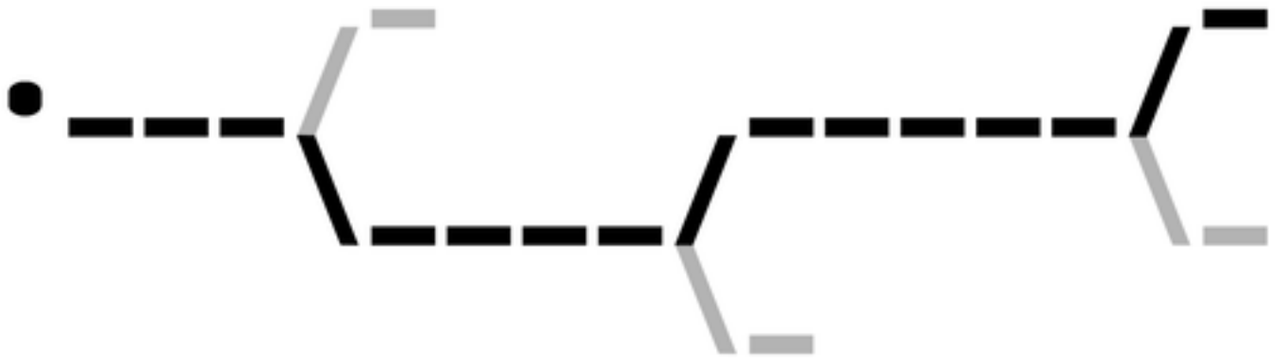
```
#define HOTCOUNT_SIZE   64

hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]
```

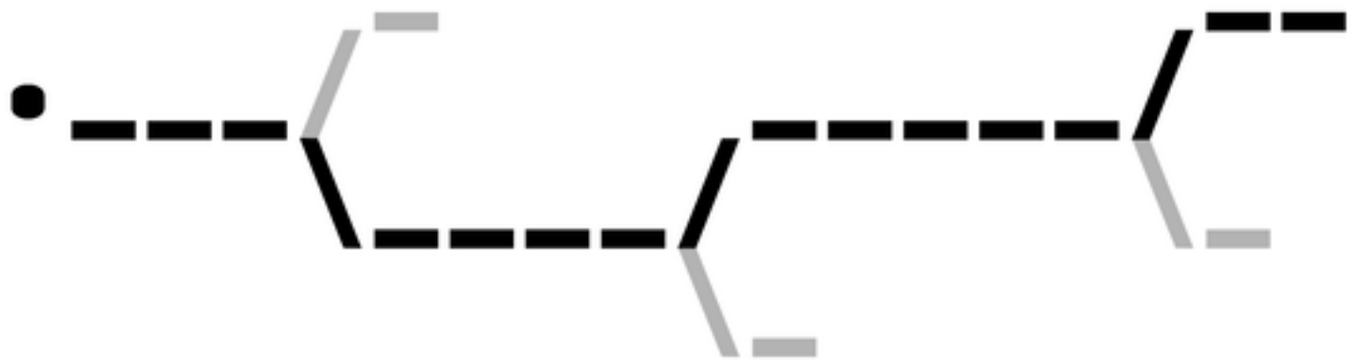
```
#define HOTCOUNT_SIZE   64

hotcount[(PC>>2) & (HOTCOUNT_SIZE-1)]

/* can cause non-determenism */
```

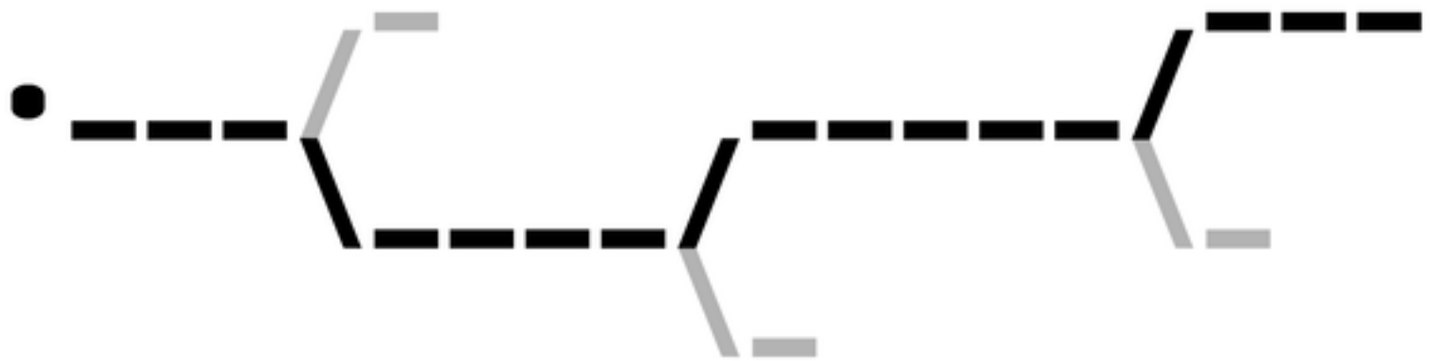
recording pipeline

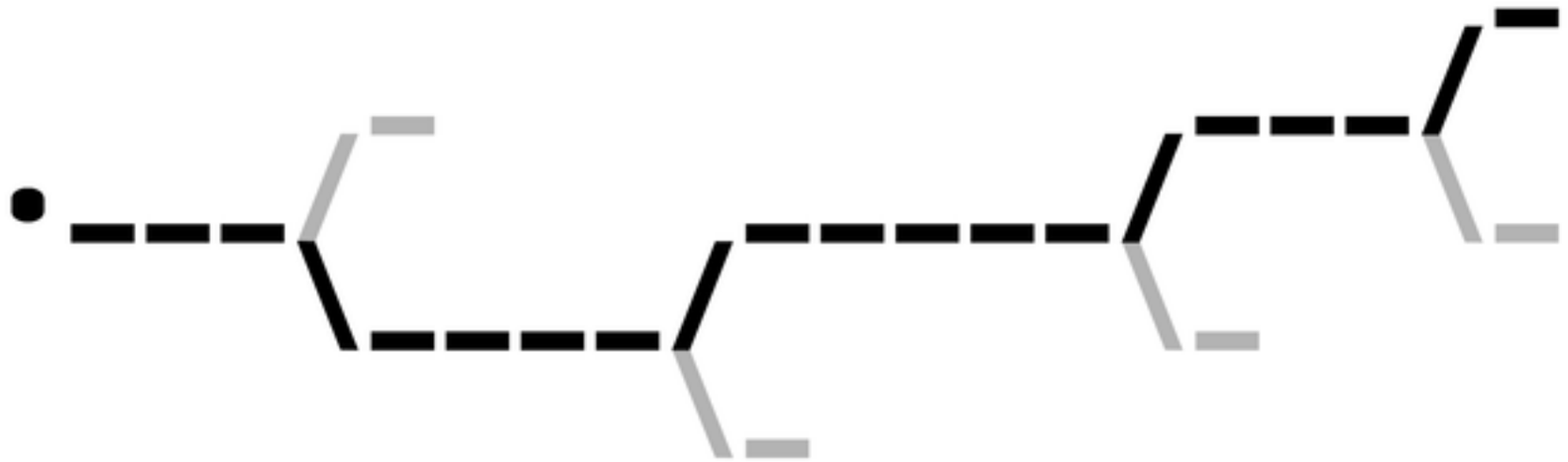
tracing 101

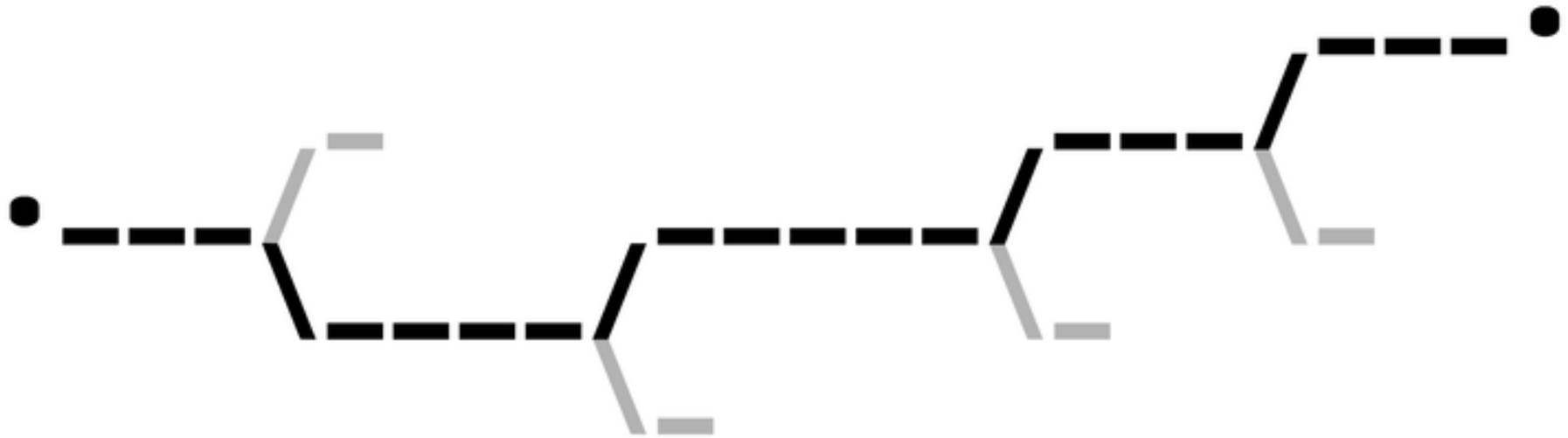










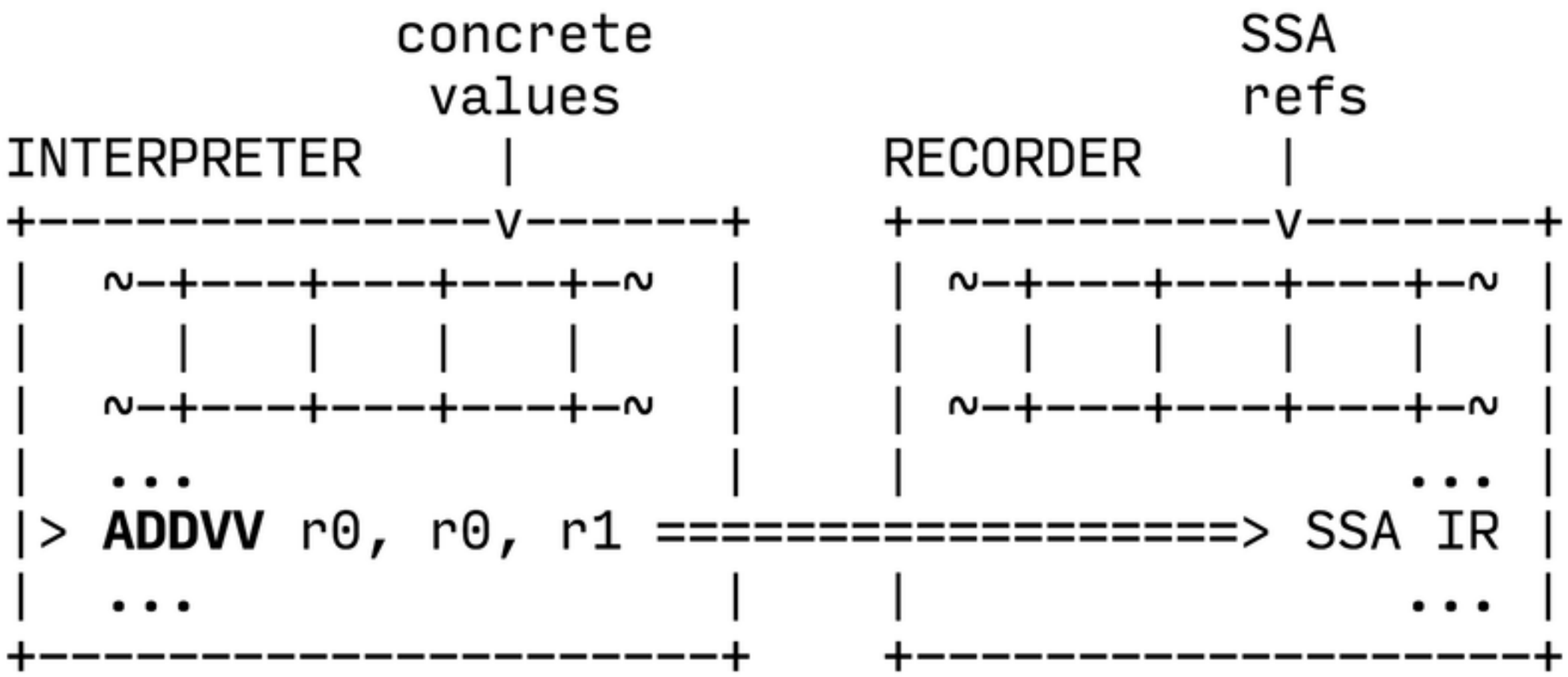


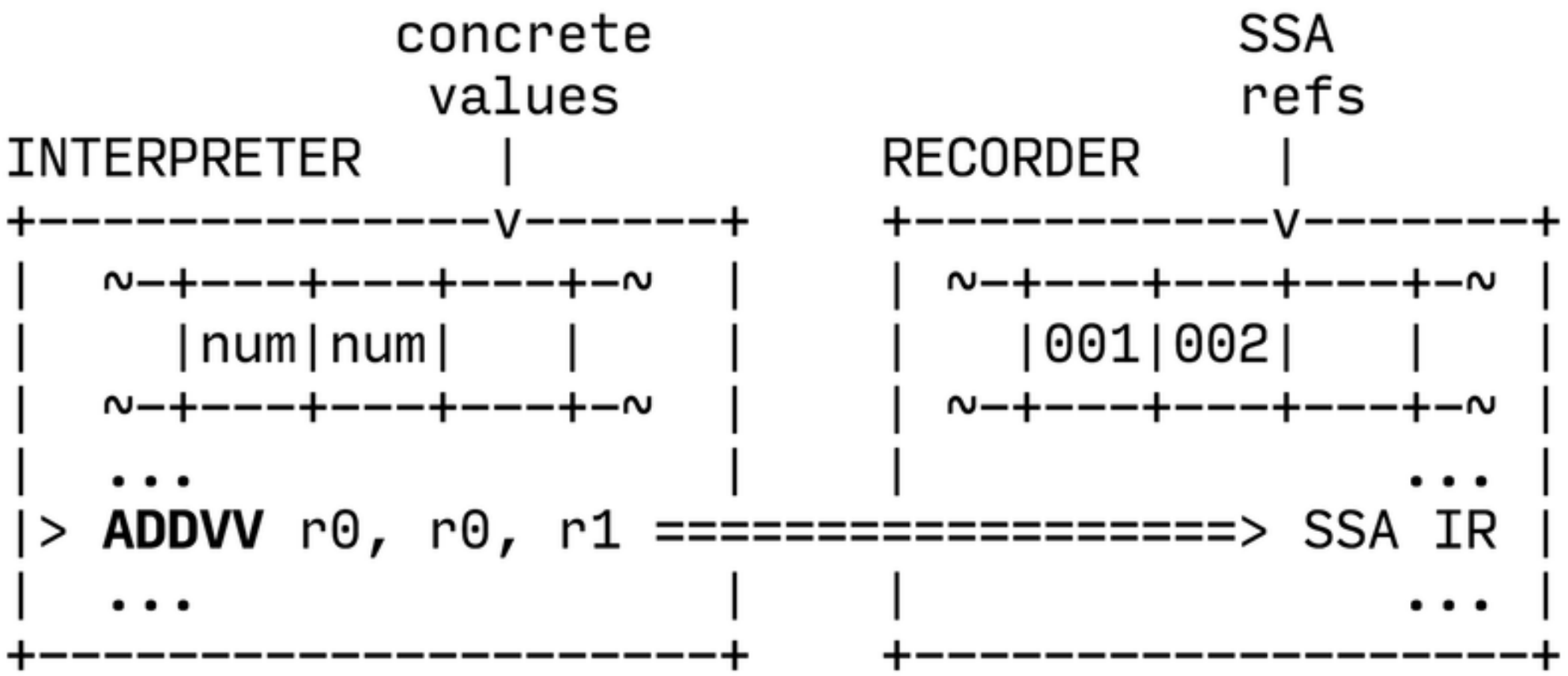

guard

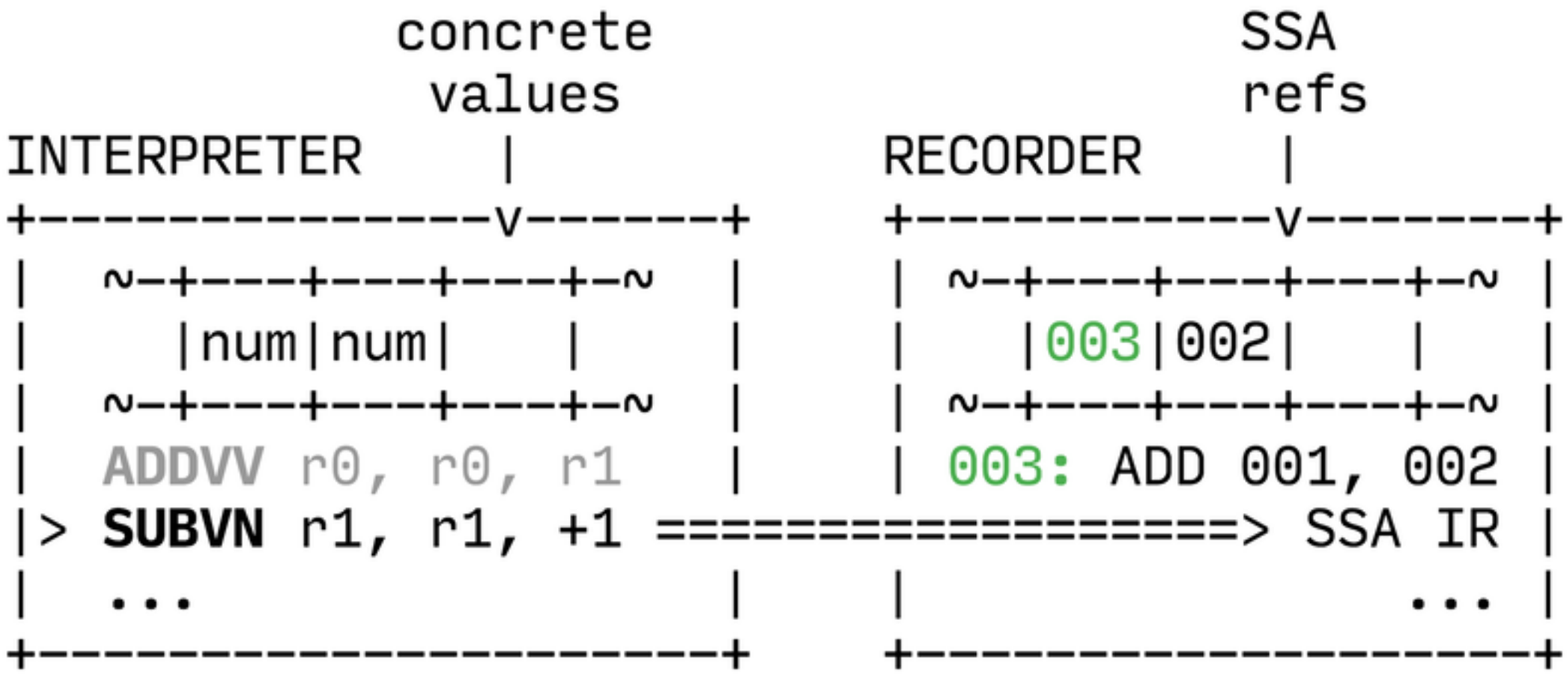
hot side exits spawn side traces

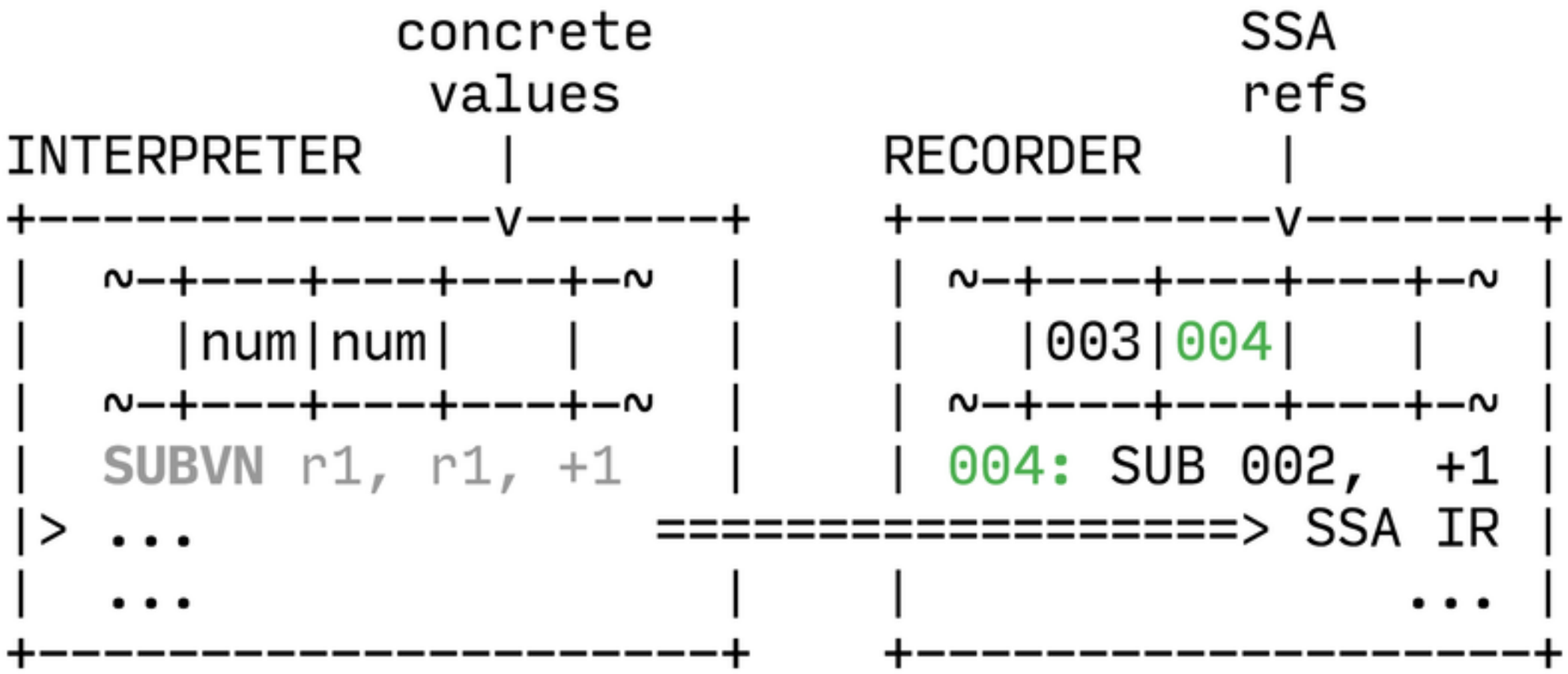
.___☠ (trace abort)

back to recording


concrete
values
SSA
refs
INTERPRETER
RECORDER
> ADDVV r0, r0, r1
SSA IR


concrete
values
SSA
refs
INTERPRETER
RECORDER
num num
001 002
> ADDVV r0, r0, r1
SSA IR


concrete
values
SSA
refs
INTERPRETER
RECORDER
num num
003 002
ADDVV r0, r0, r1
003: ADD 001, 002
SUBVN r1, r1, +1
SSA IR
...
...

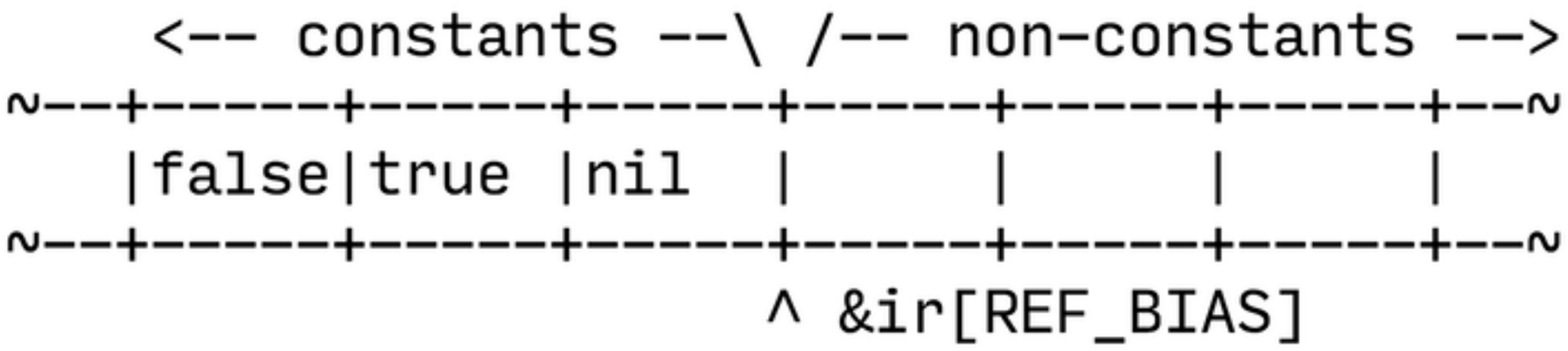

concrete
values
INTERPRETER
num num
SUBVN r1, r1, +1
...
...
SSA
refs
RECORDER
003 004
004: SUB 002, +1
SSA IR
...

IR

```
/* Trace object. */
typedef struct GCtrace {
  /* IR instructions/constants.
  ** Biased with REF_BIAS.
  */
  IRIns *ir;

} GCtrace;
```

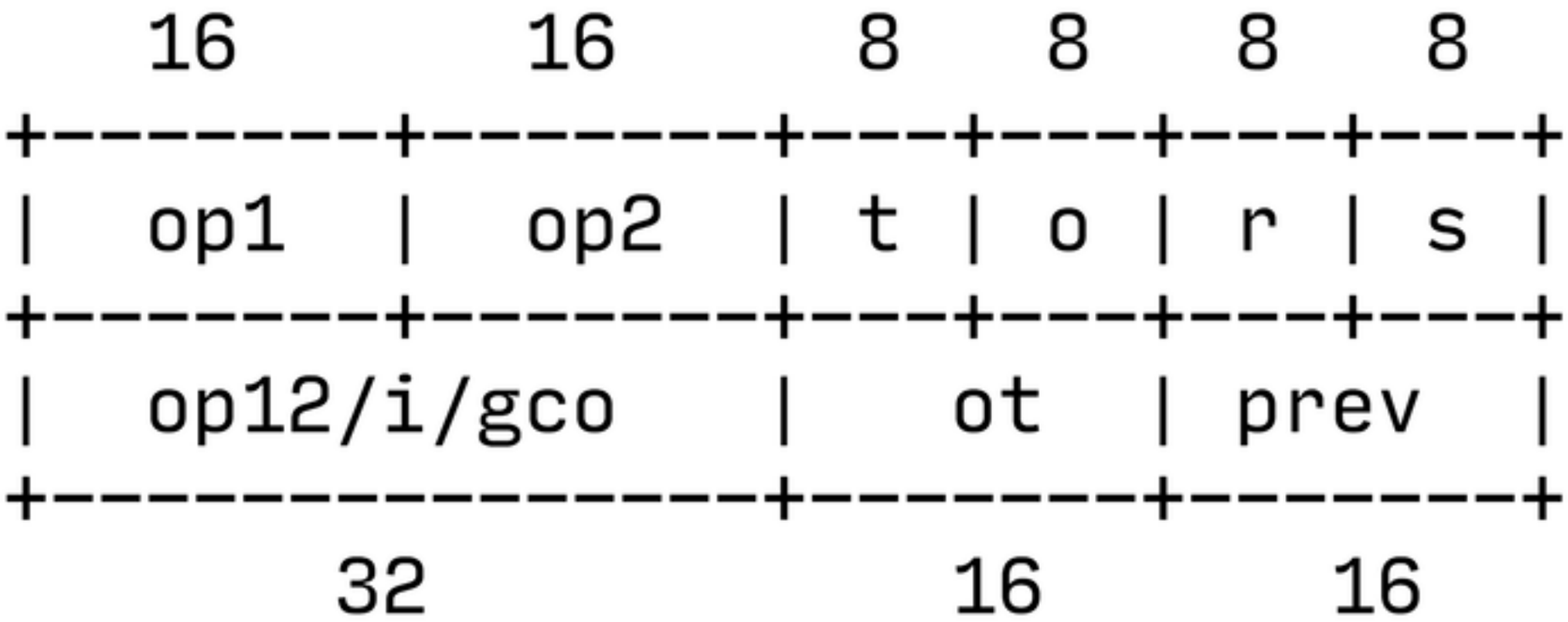
```
/* Trace object. */
typedef struct GCtrace {
  /* IR instructions/constants.
  ** Biased with REF_BIAS.
  */
  IRIns *ir;

} GCtrace;
```

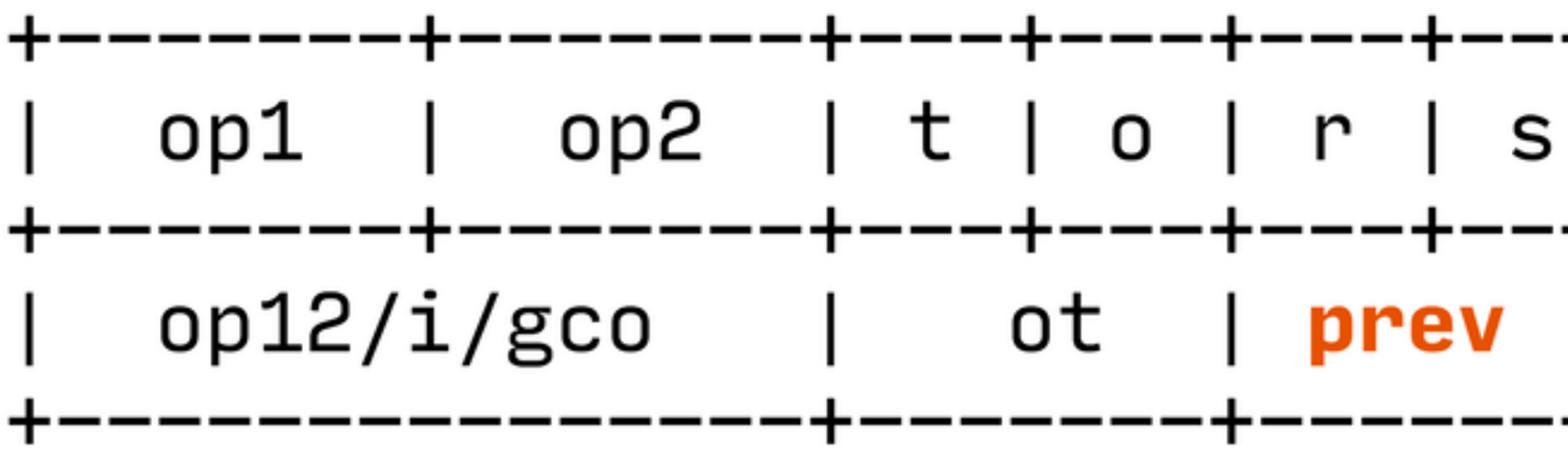
```
typedef uint16_t IRRef1;

/* Fixed references. */
enum {
  REF_TRUE =  REF_BIAS-3,
  REF_FALSE = REF_BIAS-2,
  REF_NIL = REF_BIAS-1,
  /* \--- Constants grow downwards. */
  REF_BIAS =  0x8000,
  /* /--- IR grows upwards. */
  REF_FIRST = REF_BIAS+1,
  REF_DROP =  0xffff
};
```



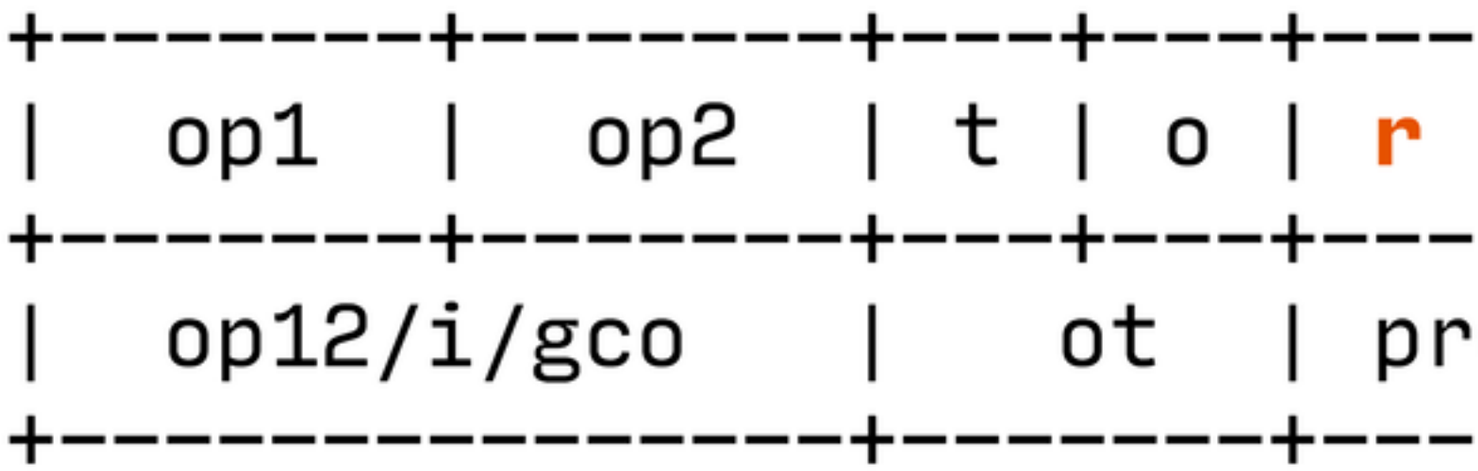

```
ir := irbuf + nconsts - REF_BIAS
```

IRIns

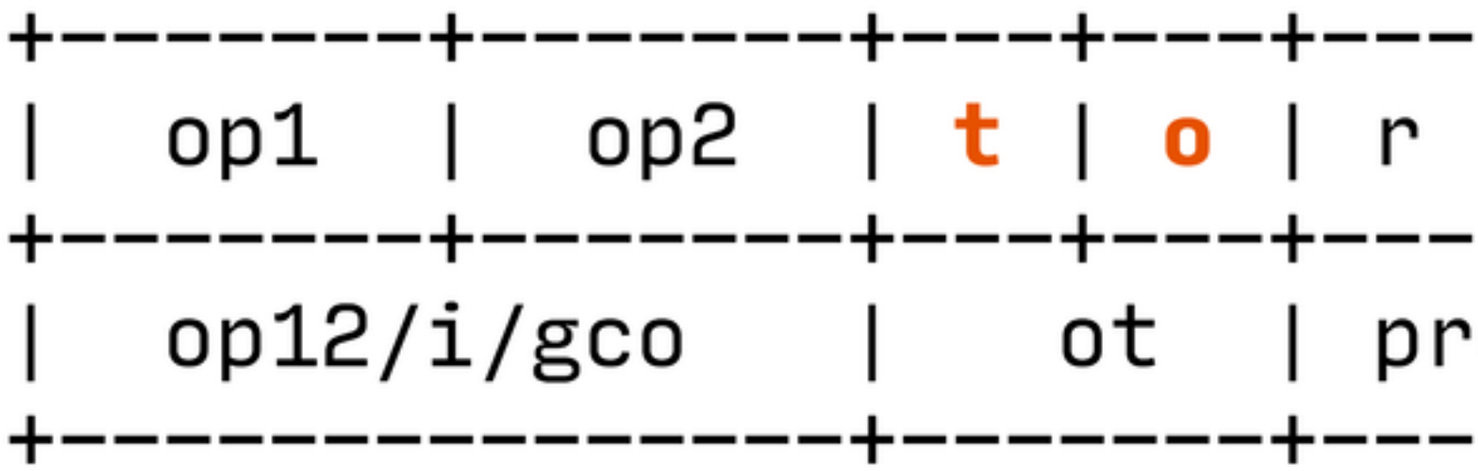




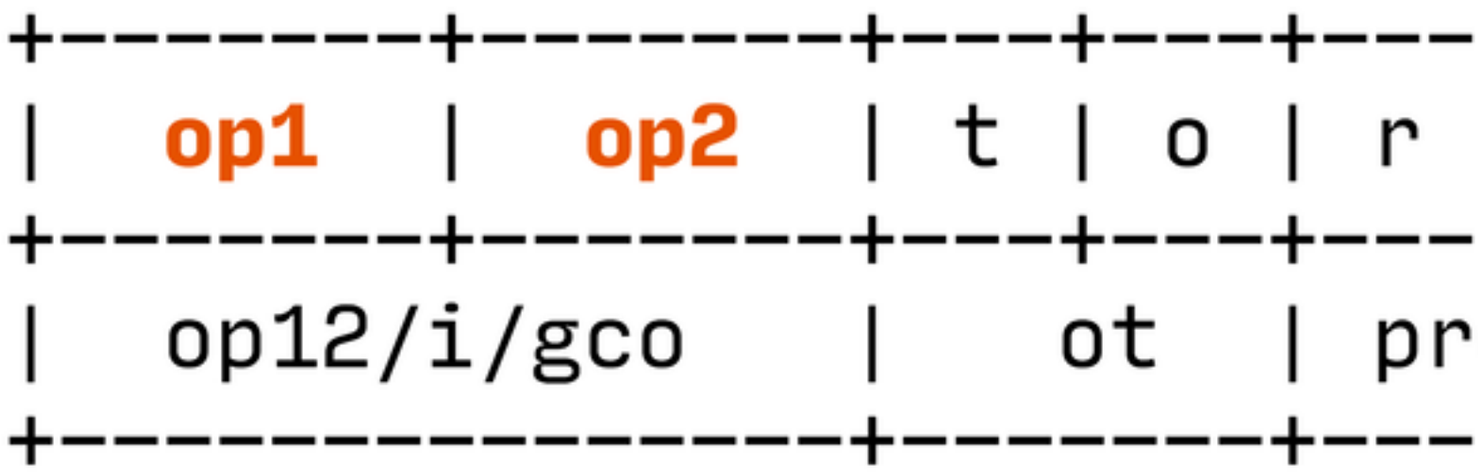
prev is the reference to the previous instruction with the same opcode



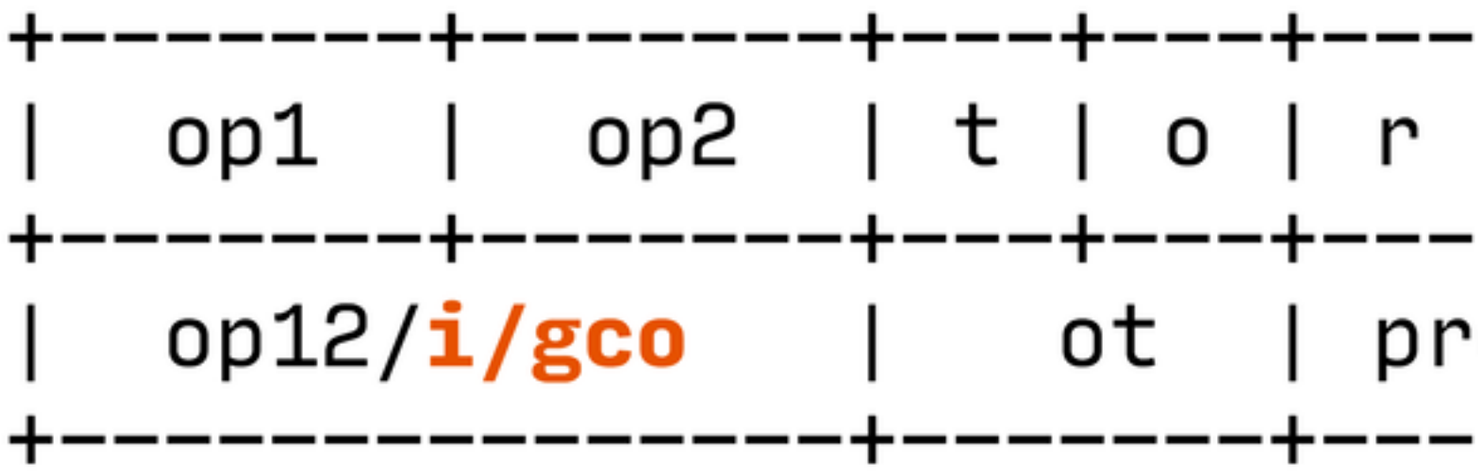
r/s register allocation state



o opcode
t type



op1/op2 IR references



i/gco constants (32 bit)

```
/* Tagged IR references (32 bit).
**
** +-------+-------+----------------+
** |  irt  | flags |       ref      |
** +-------+-------+----------------+
**
** The tag holds a copy of the IRType
** and speeds up IR type checks.
*/

typedef uint32_t TRef;
```

BYTECODE ==========> SSA IR


BYTECODE
SSA IR
interpret -> specialize -> fold&emit

```
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

```
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

```
case BC_LEN:
  if (tref_isstr(rc))
    rc = emitir(IRTI(IR_FLOAD), rc, IRFL_STR_LEN);
  else if (!LJ_52 && tref_istab(rc))
    rc = lj_ir_call(J, IRCALL_lj_tab_len, rc);
  else
    rc = rec_mm_len(J, rc, rcv);
  break;
```

`emitir` passes instruction to FOLD engine

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
// Rules hashtable generated by build
// Rules applied until fixpoint
```

FWD
DSE
NARROW
ABCelim
CSE

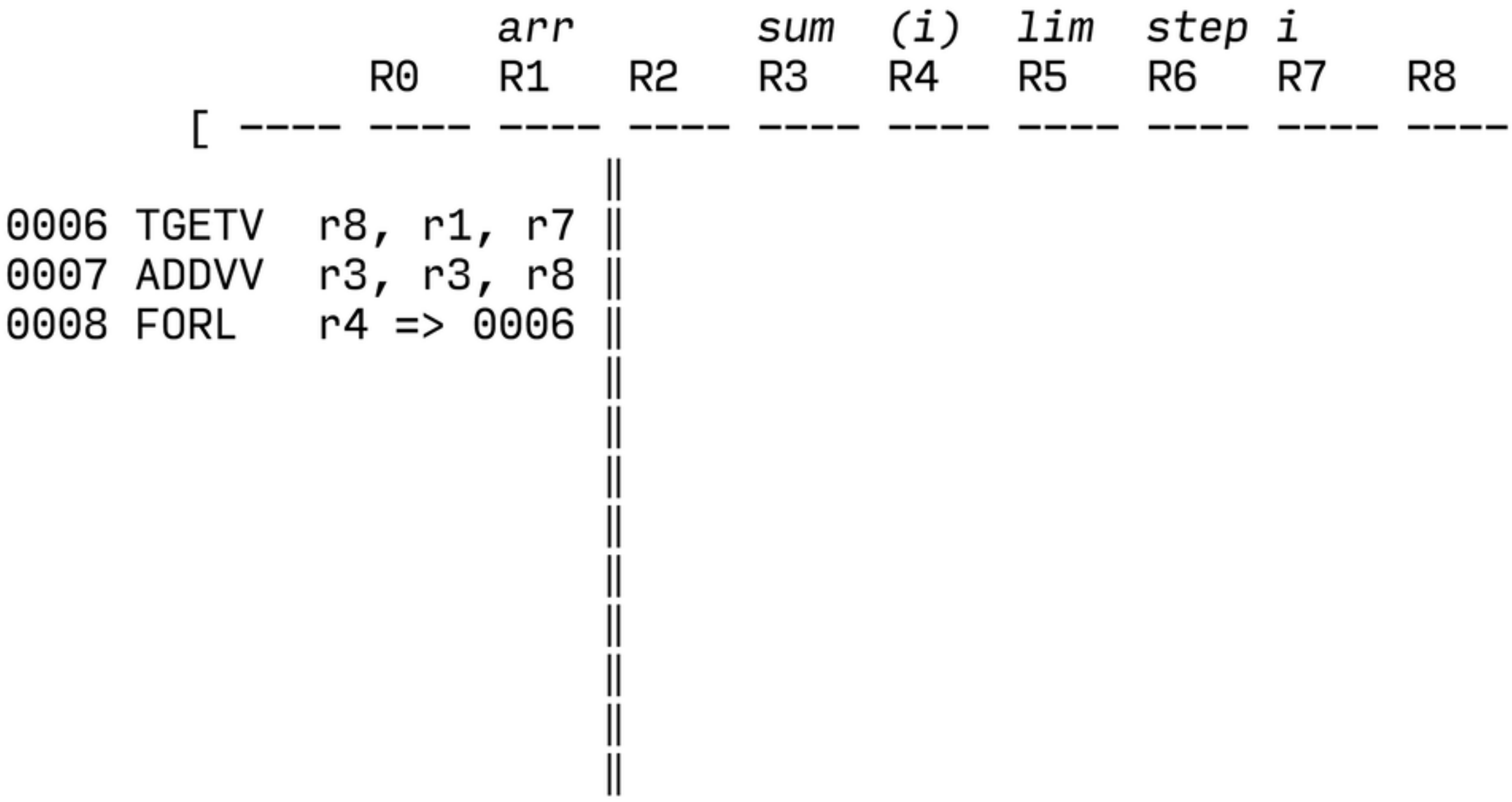
DCE
LOOP
SPLIT
SINK

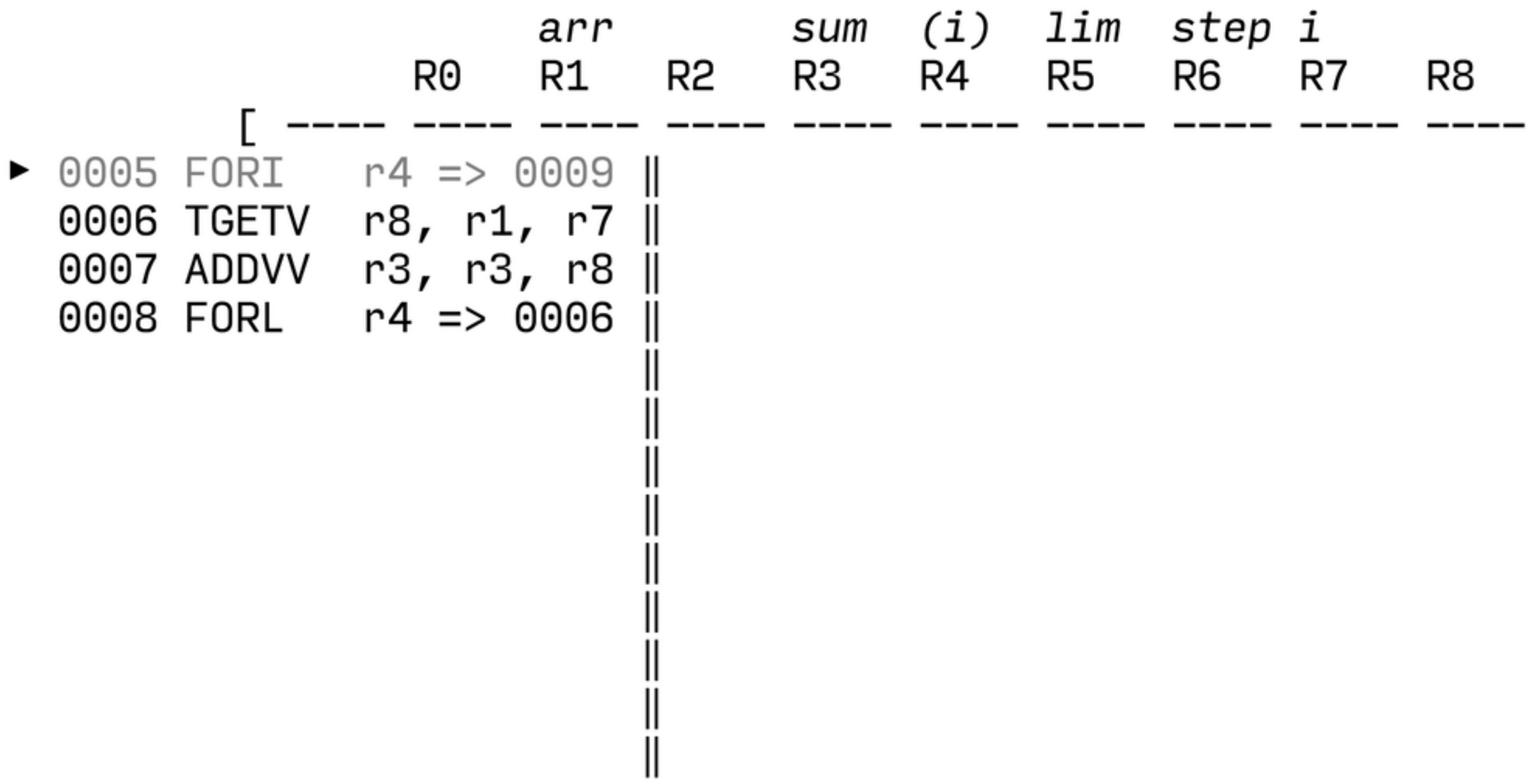
DCE
LOOP
SPLIT
SINK

```
local sum = 0
for i = 1, n do
  sum = sum + arr[i]
end
```

```
0006 TGETV   r8, r1, r7
0007 ADDVV   r3, r3, r8
0008 FORL    r4 => 0006
```

```
0006 TGETV   r8, r1, r7 ; r8 = r1[r7]
0007 ADDVV   r3, r3, r8 ; r3 = r3 + r8
0008 FORL    r4 => 0006 ; r4 = r4 + r6
                        ; if r4 <= r5 then
                        ;    r7 = r4
                        ;    jump 0006
                        ; end
```


arr sum (i) lim step i
R0 R1 R2 R3 R4 R5 R6 R7 R8
[---- ---- ---- ---- ---- ---- ---- ---- ---- ----
0006 TGETV r8, r1, r7
0007 ADDVV r3, r3, r8
0008 FORL r4 => 0006


arr sum (i) lim step i
R0 R1 R2 R3 R4 R5 R6 R7 R8
0005 FORI r4 => 0009
0006 TGETV r8, r1, r7
0007 ADDVV r3, r3, r8
0008 FORL r4 => 0006

```
                        arr             sum  (i)  lim  step i
                  R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  ----  ----  ----  0003  0001  ----  0003  ----
▶ 0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                        arr         sum  (i)  lim  step i
                 R0     R1    R2    R3    R4    R5    R6    R7    R8
          [ ---- ----  0004 ----  ----  0003 0001 ----  0003 ----
  0005 FORI    r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV   r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV   r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL    r4 => 0006 ‖ 0004 SLOAD  R1
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
                          ‖
```

```
                                arr             sum  (i)  lim  step i
                    R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  0004  ----  ----  0003  0001  ----  0003  ----
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

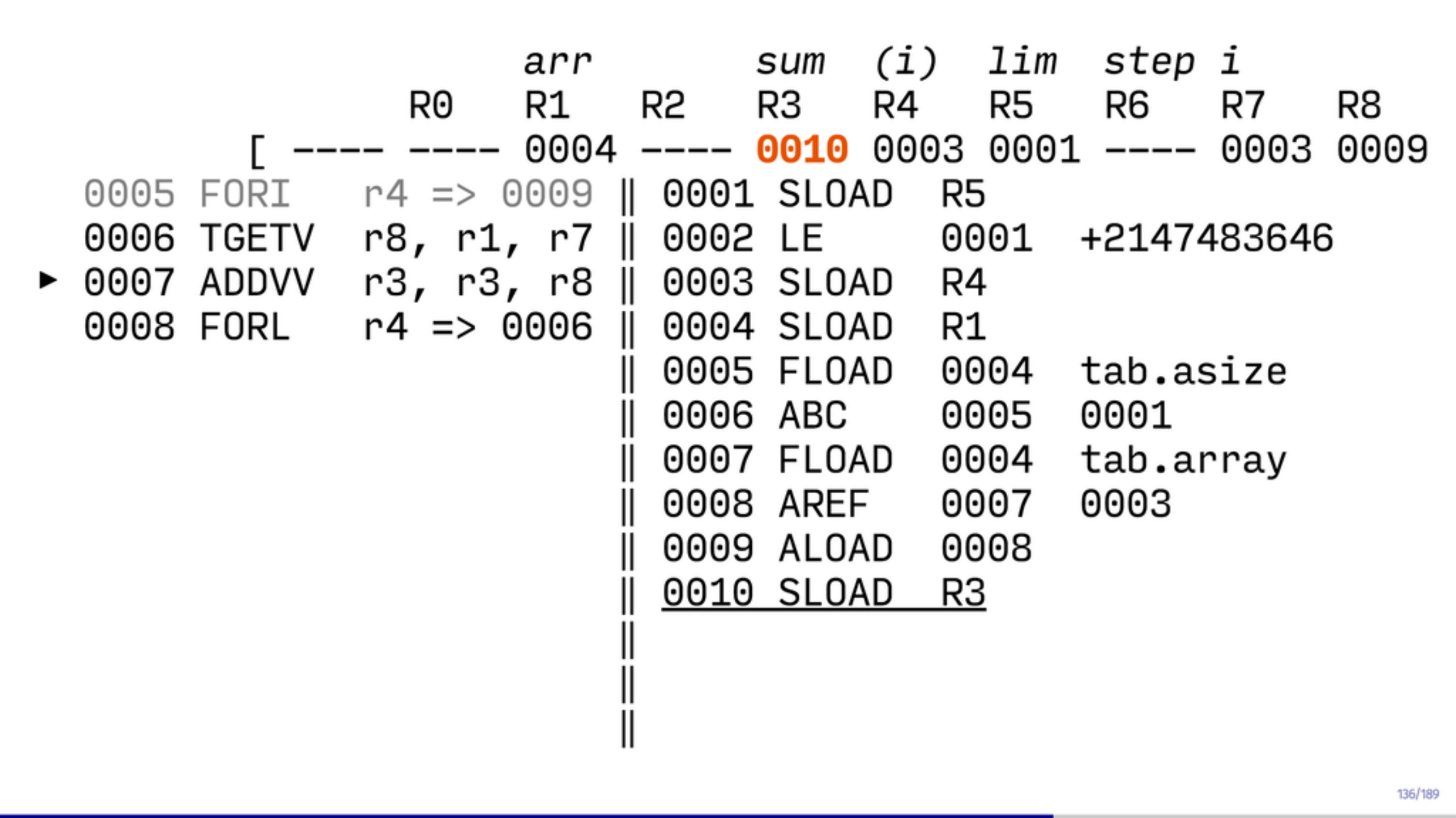
```
                                arr             sum  (i)  lim  step i
                    R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  0004  ----  ----  0003  0001  ----  0003  ----
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

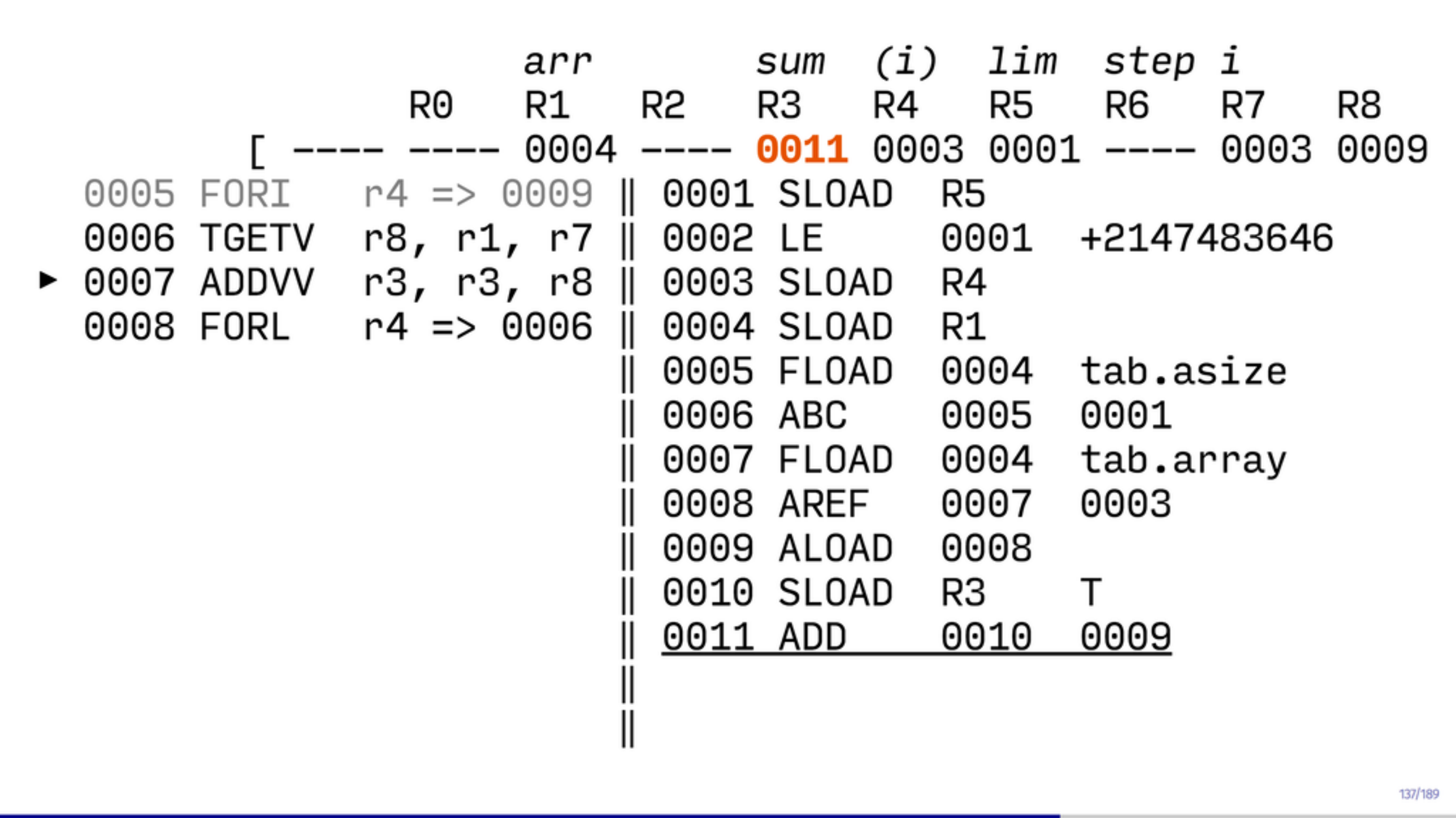
```
                                 arr             sum  (i)  lim  step i
                    R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  0004  ----  ----  0003  0001  ----  0003  ----
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                          arr          sum  (i)  lim  step i
                 R0   R1   R2   R3   R4   R5   R6   R7   R8
          [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 ----
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

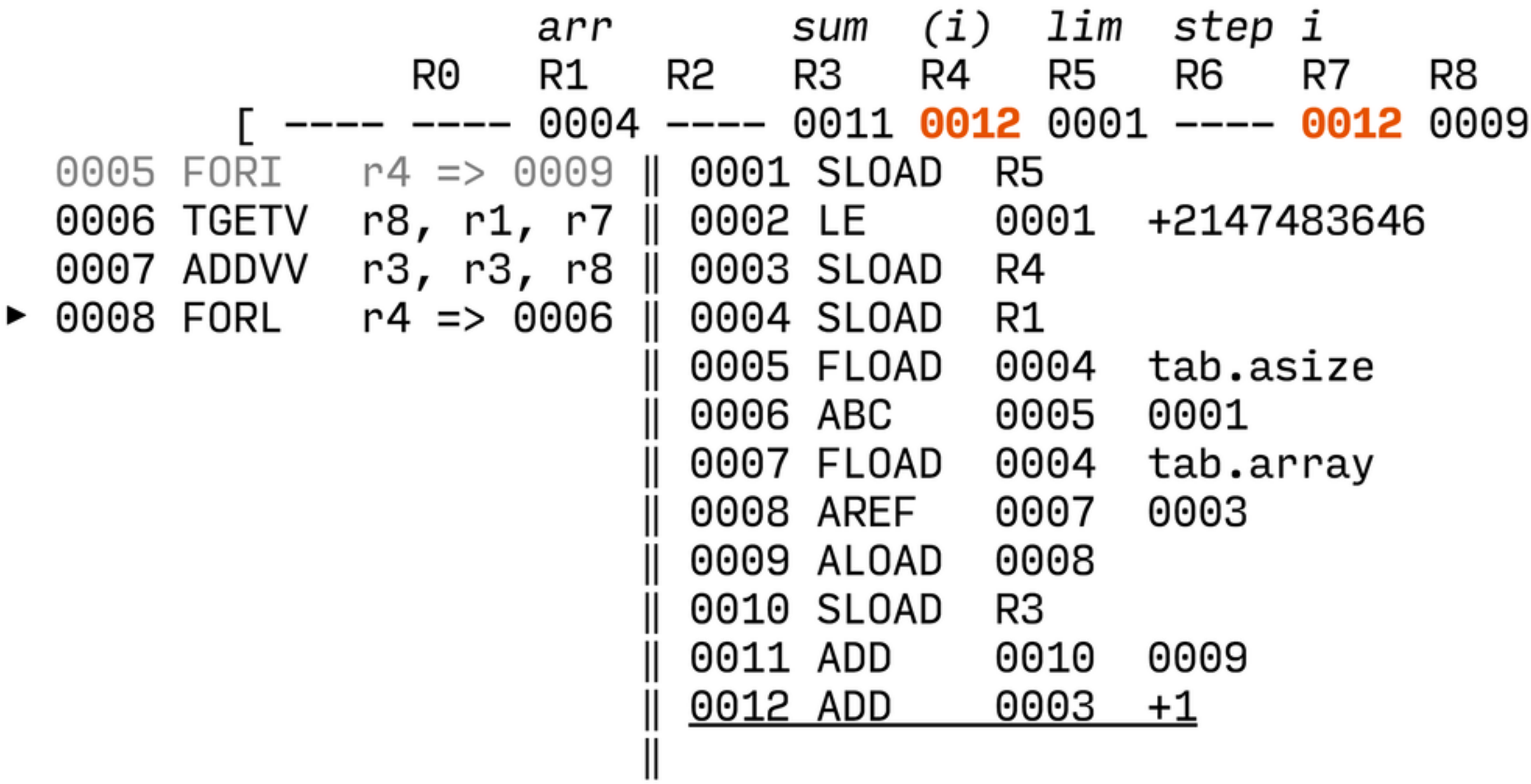
```
                          arr          sum  (i)  lim  step i
                 R0   R1   R2   R3   R4   R5   R6   R7   R8
          [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 0009
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
▶ 0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖
                         ‖
                         ‖
                         ‖
                         ‖
```

```
                 arr        sum  (i)  lim  step i
            R0   R1   R2   R3   R4   R5   R6   R7   R8
       [ ---- ---- 0004 ---- ---- 0003 0001 ---- 0003 0009
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
► 0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
  0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖
                         ‖
                         ‖
                         ‖
```


arr sum (i) lim step i
R0 R1 R2 R3 R4 R5 R6 R7 R8
[---- ---- 0004 ---- 0010 0003 0001 ---- 0003 0009
0005 FORI r4 => 0009
0006 TGETV r8, r1, r7
0007 ADDVV r3, r3, r8
0008 FORL r4 => 0006
0001 SLOAD R5
0002 LE 0001 +2147483646
0003 SLOAD R4
0004 SLOAD R1
0005 FLOAD 0004 tab.asize
0006 ABC 0005 0001
0007 FLOAD 0004 tab.array
0008 AREF 0007 0003
0009 ALOAD 0008
0010 SLOAD R3


arr sum (i) lim step i
R0 R1 R2 R3 R4 R5 R6 R7 R8
[---- ---- 0004 ---- 0011 0003 0001 ---- 0003 0009
0005 FORI r4 => 0009
0006 TGETV r8, r1, r7
0007 ADDVV r3, r3, r8
0008 FORL r4 => 0006
0001 SLOAD R5
0002 LE 0001 +2147483646
0003 SLOAD R4
0004 SLOAD R1
0005 FLOAD 0004 tab.asize
0006 ABC 0005 0001
0007 FLOAD 0004 tab.array
0008 AREF 0007 0003
0009 ALOAD 0008
0010 SLOAD R3 T
0011 ADD 0010 0009

```
                                arr          sum  (i)  lim  step i
                    R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  0004  ----  0011  0003  0001  ----  0003  0009
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
▶ 0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖ 0010 SLOAD  R3
                         ‖ 0011 ADD    0010  0009
                         ‖
                         ‖
```


arr sum (i) lim step i
R0 R1 R2 R3 R4 R5 R6 R7 R8
[---- ---- 0004 ---- 0011 0012 0001 ---- 0012 0009
0005 FORI r4 => 0009
0006 TGETV r8, r1, r7
0007 ADDVV r3, r3, r8
▶ 0008 FORL r4 => 0006
0001 SLOAD R5
0002 LE 0001 +2147483646
0003 SLOAD R4
0004 SLOAD R1
0005 FLOAD 0004 tab.asize
0006 ABC 0005 0001
0007 FLOAD 0004 tab.array
0008 AREF 0007 0003
0009 ALOAD 0008
0010 SLOAD R3
0011 ADD 0010 0009
0012 ADD 0003 +1

```
                          arr          sum  (i)  lim  step i
                  R0    R1    R2    R3    R4    R5    R6    R7    R8
          [ ----  ----  0004  ----  0011  0012  0001  ----  0012  0009
  0005 FORI   r4 => 0009 ‖ 0001 SLOAD  R5
  0006 TGETV  r8, r1, r7 ‖ 0002 LE     0001  +2147483646
  0007 ADDVV  r3, r3, r8 ‖ 0003 SLOAD  R4
▶ 0008 FORL   r4 => 0006 ‖ 0004 SLOAD  R1
                         ‖ 0005 FLOAD  0004  tab.asize
                         ‖ 0006 ABC    0005  0001
                         ‖ 0007 FLOAD  0004  tab.array
                         ‖ 0008 AREF   0007  0003
                         ‖ 0009 ALOAD  0008
                         ‖ 0010 SLOAD  R3
                         ‖ 0011 ADD    0010  0009
                         ‖ 0012 ADD    0003  +1
                         ‖ 0013 LE     0012  0001
```

```
0001 >   int SLOAD  #6    CRI
0002 >   int LE     0001  +2147483646
0003     int SLOAD  #5    CI
0004 >   tab SLOAD  #2    T
0005     int FLOAD  0004  tab.asize
0006 >   p32 ABC    0005  0001
0007     p32 FLOAD  0004  tab.array
0008     p32 AREF   0007  0003
0009 >   num ALOAD  0008
0010 >   num SLOAD  #4    T
0011  +  num ADD    0010  0009
0012  +  int ADD    0003  +1
0013 >   int LE     0012  0001
....          SNAP   #2   [ - - - - 0011 0012 0001 - 0012 ]
```

```
0001 >   int SLOAD   #6    CRI
0002 >   int LE      0001  +2147483646
0003     int SLOAD   #5    CI
0004 >   tab SLOAD   #2    T
0005     int FLOAD   0004  tab.asize
0006 >   p32 ABC     0005  0001
0007     p32 FLOAD   0004  tab.array
0008     p32 AREF    0007  0003
0009 >   num ALOAD   0008
0010 >   num SLOAD   #4    T
0011  +  num ADD     0010  0009
0012  +  int ADD     0003  +1
0013 >   int LE      0012  0001
....             SNAP   #2   [ - - - - 0011 0012 0001 - 0012 ]
```

```
0001 SLOAD  #6    CRI             |
0002 LE     0001  +2147483646     |
0003 SLOAD  #5    CI              |
0004 SLOAD  #2    T               |
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             |
0002 LE     0001  +2147483646     |
0003 SLOAD  #5    CI              |
0004 SLOAD  #2    T               |
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI              | ===> 0001
0002 LE     0001  +2147483646      |
0003 SLOAD  #5    CI               |
0004 SLOAD  #2    T                |
0005 FLOAD  0004  tab.asize        |
0006 ABC    0005  0001             |
0007 FLOAD  0004  tab.array        |
0008 AREF   0007  0003             |
0009 ALOAD  0008                   |
0010 SLOAD  #4    T                |
0011 ADD    0010  0009             |
0012 ADD    0003  +1               |
0013 LE     0012  0001             |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     |
0003 SLOAD  #5    CI              |
0004 SLOAD  #2    T               |
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI              | 0001
0002 LE     0001  +2147483646      | ===> LE    [0001] +214748364
0003 SLOAD  #5    CI               |
0004 SLOAD  #2    T                |
0005 FLOAD  0004  tab.asize        |
0006 ABC    0005  0001             |
0007 FLOAD  0004  tab.array        |
0008 AREF   0007  0003             |
0009 ALOAD  0008                   |
0010 SLOAD  #4    T                |
0011 ADD    0010  0009             |
0012 ADD    0003  +1               |
0013 LE     0012  0001             |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI              | 0001
0002 LE     0001  +2147483646      | ===> LE     0001  +214748364
0003 SLOAD  #5    CI               |
0004 SLOAD  #2    T                |
0005 FLOAD  0004  tab.asize        |
0006 ABC    0005  0001             |
0007 FLOAD  0004  tab.array        |
0008 AREF   0007  0003             |
0009 ALOAD  0008                   |
0010 SLOAD  #4    T                |
0011 ADD    0010  0009             |
0012 ADD    0003  +1               |
0013 LE     0012  0001             |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              |
0004 SLOAD  #2    T               |
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               |
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       |
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | ===> FLOAD  0004  tab.asize
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            |
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | ===> ABC    0005  0001
0007 FLOAD  0004  tab.array      |
0008 AREF   0007  0003           |
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            | 0006
0007 FLOAD  0004  tab.array       |
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            | 0006
0007 FLOAD  0004  tab.array       | 0007
0008 AREF   0007  0003            |
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | ===> AREF  [0007][0003]
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            | 0006
0007 FLOAD  0004  tab.array       | 0007
0008 AREF   0007  0003            | ===> AREF   0007  0012
0009 ALOAD  0008                  |
0010 SLOAD  #4    T               |
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 |
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              |
0011 ADD    0010  0009           |
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            | 0006
0007 FLOAD  0004  tab.array       | 0007
0008 AREF   0007  0003            | 0015 AREF   0007  0012
0009 ALOAD  0008                  | 0016 ALOAD  0015
0010 SLOAD  #4    T               | 0011
0011 ADD    0010  0009            |
0012 ADD    0003  +1              |
0013 LE     0012  0001            |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             |
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             | 0018 ADD    0012  +1
0013 LE     0012  0001           |
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             | 0018 ADD    0012  +1
0013 LE     0012  0001           | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             | 0018 ADD    0012  +1
0013 LE     0012  0001           | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
```

```
0001 SLOAD  #6    CRI            | 0001
0002 LE     0001  +2147483646    | 0002
0003 SLOAD  #5    CI             | 0012
0004 SLOAD  #2    T              | 0004
0005 FLOAD  0004  tab.asize      | 0005
0006 ABC    0005  0001           | 0006
0007 FLOAD  0004  tab.array      | 0007
0008 AREF   0007  0003           | 0015 AREF   0007  0012
0009 ALOAD  0008                 | 0016 ALOAD  0015
0010 SLOAD  #4    T              | 0011
0011 ADD    0010  0009           | 0017 ADD    0011  0016
0012 ADD    0003  +1             | 0018 ADD    0012  +1
0013 LE     0012  0001           | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
.... SNAP   [ ---- ---- ---- ---- 0017 0018 0001 ---- 0018 ]
```

```
0001 SLOAD  #6    CRI             | 0001
0002 LE     0001  +2147483646     | 0002
0003 SLOAD  #5    CI              | 0012
0004 SLOAD  #2    T               | 0004
0005 FLOAD  0004  tab.asize       | 0005
0006 ABC    0005  0001            | 0006
0007 FLOAD  0004  tab.array       | 0007
0008 AREF   0007  0003            | 0015 AREF   0007  0012
0009 ALOAD  0008                  | 0016 ALOAD  0015
0010 SLOAD  #4    T               | 0011
0011 ADD    0010  0009            | 0017 ADD    0011  0016
0012 ADD    0003  +1              | 0018 ADD    0012  +1
0013 LE     0012  0001            | 0019 LE     0018  0001
.... SNAP   [ ---- ---- ---- ---- 0011 0012 0001 ---- 0012 ]
.... SNAP   [ ---- ---- ---- ---- 0017 0018 0001 ---- 0018 ]
```

```
0001 SLOAD  #6    CRI           | 0001
0002 LE     0001  +2147483646   | 0002
0003 SLOAD  #5    CI            | 0012
0004 SLOAD  #2    T             | 0004
0005 FLOAD  0004  tab.asize     | 0005
0006 ABC    0005  0001          | 0006
0007 FLOAD  0004  tab.array     | 0007
0008 AREF   0007  0003          | 0015 AREF   0007  0012
0009 ALOAD  0008                | 0016 ALOAD  0015
0010 SLOAD  #4    T             | 0011
0011 ADD    0010  0009          | 0017 ADD    0011  0016
0012 ADD    0003  +1            | 0018 ADD    0012  +1
0013 LE     0012  0001          | 0019 LE     0018  0001
                                  0020 PHI    0012  0018
                                  0021 PHI    0011  0017
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */

  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  /* What if fleft is not invariant? */
  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  PHIBARRIER(fleft);
  return fleft->op2;
}
```

```
LJFOLD(FLOAD SNEW IRFL_STR_LEN)
LJFOLDF(fload_str_len_snew)
{
  /* Return length passed to SNEW. */
  PHIBARRIER(fleft);
  return fleft->op2;
}
```

DCE
LOOP
SPLIT
SINK

assemble

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
```

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
/* looks a bit strange?  */
```

```
asm_guardcc(as, CC_E);
emit_rr(as, XO_TEST, RID_RET, RID_RET);
/* assembled backwards!  */
/* test rax, rax; je ... */
```

linear scan

THE END

What I learned from LuaJIT

ELEGANCE IS A DOUBLE-EDGED SWORD

DO NOT FEAR
THE PREPROCESSING

USERS DON'T UNDER-STAND

WHAT IS FAST

PERFORMANCE IMPLI-CATIONS OF TRACING ARE NONTRIVIAL

SEARCH FOR THE BALANCE

MAKE YOUR OWN RULES

THANK YOU!