Optimizations

Do no harm, i.e., do not change the meaning of the program in any way

Changing a 3 to a 4 is obviously bad, but replacing error-signalling code with code that does not signal the error *is* harm.

Optimizations

Consider a <*insert life-critical device with software here*> that catches such errors and goes into a fallback, safe state

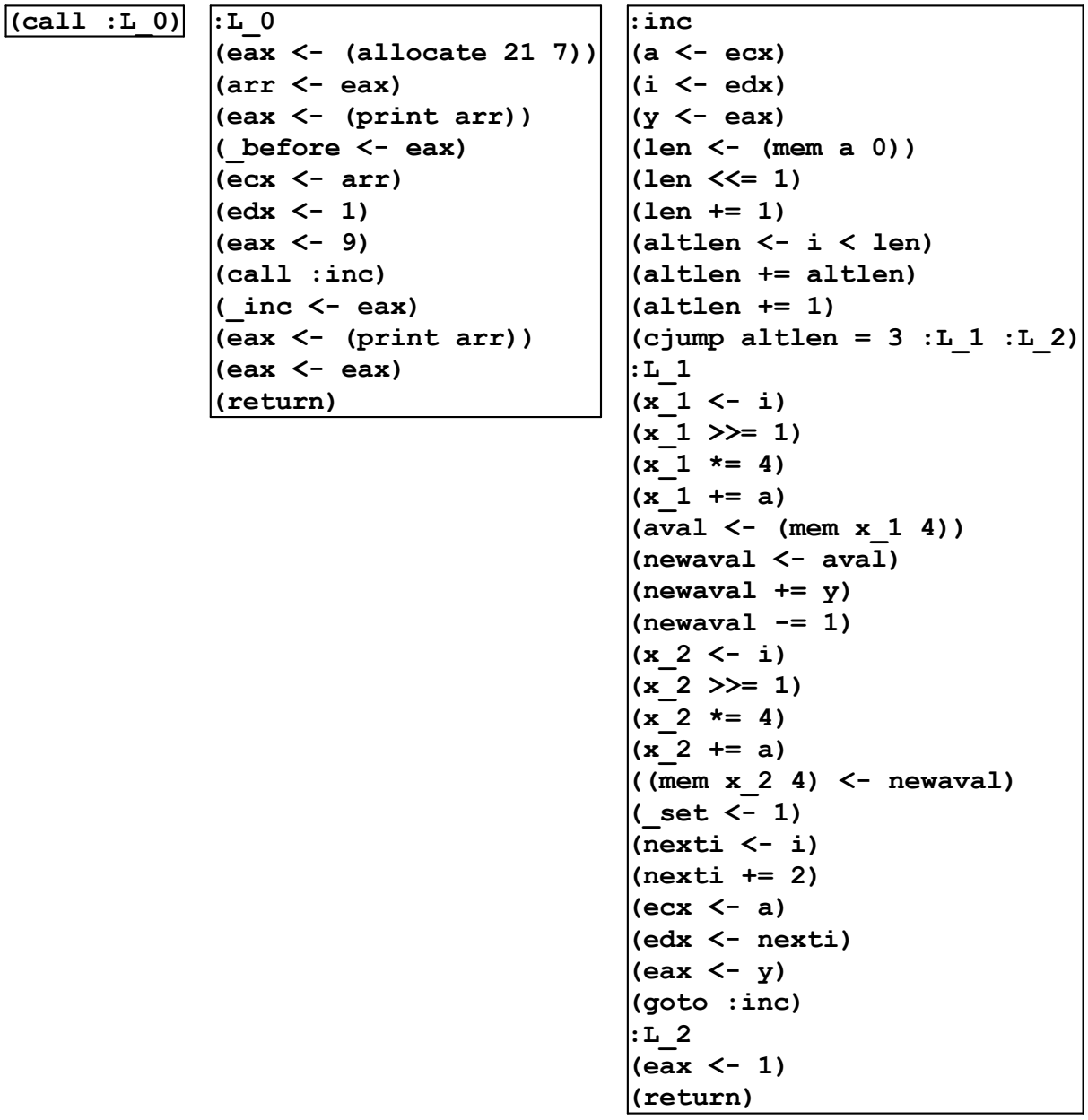
- When the pacemaker's software has an error; catch the error, stop stimulating the heart, and call 911
- When the autopilot software has an error; catch the error and wake up the pilot
- When the train crossing software crashes, catch the error and block all cars from passing

Consider this example L3 program

```
((let ((arr (new-array 10 3)))
   (let ((_before (print arr)))
     (let ((_inc (:inc arr 0 4)))
       (print arr))))
 (:inc
  (a i y)
  (let ((len (alen a)))
    (let ((altlen (< i len)))
      (if altlen
        (let ((aval (aref a i)))
          (let ((newaval (+ aval y)))
            (let ((_set (aset a i newaval)))
              (let ((nexti (+ i 1)))
                (:inc a nexti y)))))
        0)))))
```

Here is its compilation into L2

(note how **`main`** is compiled)


(call :L_0)
:L_0
(eax <- (allocate 21 7))
(arr <- eax)
(eax <- (print arr))
(_before <- eax)
(ecx <- arr)
(edx <- 1)
(eax <- 9)
(call :inc)
(_inc <- eax)
(eax <- (print arr))
(eax <- eax)
(return)
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(newaval += y)
(newaval -= 1)
(x_2 <- i)
(x_2 >>= 1)
(x_2 *= 4)
(x_2 += a)
((mem x_2 4) <- newaval)
(_set <- 1)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)

```
(:inc
 (a i y)
 (let ((len (alen a)))
   (let ((altlen (< i len)))
     (if altlen
       (let ((aval (aref a i)))
         (let ((newaval (+ aval y)))
           (let ((_set (aset a i newaval)))
             (let ((nexti (+ i 1)))
               (:inc a nexti y)))))
       0))))
```

```
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(newaval += y)
(newaval -= 1)
(x_2 <- i)
(x_2 >>= 1)
(x_2 *= 4)
(x_2 += a)
((mem x_2 4) <- newaval)
(_set <- 1)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)
```

Analysis ⇒ Optimizatoin

Run an analysis to determine some information about the program

Use that information to transform the program into (hopefully) a better one

Dead code elimination

We can use liveness analysis to identify useless code

Remove any instruction that assigns to a variable, if the variable is not in the out set of that instruction

		in	out
1:	`:inc`	()	()
2:	`(a <- ecx)`	()	()
3:	`(i <- edx)`	()	()
4:	`(y <- eax)`	()	()
5:	`(len <- (mem a 0))`	()	()
6:	`(len <<= 1)`	()	()
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	()
14:	`(x_1 >>= 1)`	()	()
15:	`(x_1 *= 4)`	()	()
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	()
19:	`(newaval += y)`	()	()
20:	`(newaval -= 1)`	()	()
21:	`(x_2 <- i)`	()	()
22:	`(x_2 >>= 1)`	()	()
23:	`(x_2 *= 4)`	()	()
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(_set <- 1)`	()	()
27:	`(nexti <- i)`	()	()
28:	`(nexti += 2)`	()	()
29:	`(ecx <- a)`	()	()
30:	`(edx <- nexti)`	()	()
31:	`(eax <- y)`	()	()
32:	`(goto :inc)`	()	()
33:	`:L_2`	()	()
34:	`(eax <- 1)`	()	()
35:	`(return)`	()	()

		in	out
1:	`:inc`	()	()
2:	`(a <- ecx)`	(ecx)	()
3:	`(i <- edx)`	(edx)	()
4:	`(y <- eax)`	(eax)	()
5:	`(len <- (mem a 0))`	(a)	()
6:	`(len <<= 1)`	(len)	()
7:	`(len += 1)`	(len)	()
8:	`(altlen <- i < len)`	(i len)	()
9:	`(altlen += altlen)`	(altlen)	()
10:	`(altlen += 1)`	(altlen)	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen)	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	(i)	()
14:	`(x_1 >>= 1)`	(x_1)	()
15:	`(x_1 *= 4)`	(x_1)	()
16:	`(x_1 += a)`	(a x_1)	()
17:	`(aval <- (mem x_1 4))`	(x_1)	()
18:	`(newaval <- aval)`	(aval)	()
19:	`(newaval += y)`	(newaval y)	()
20:	`(newaval -= 1)`	(newaval)	()
21:	`(x_2 <- i)`	(i)	()
22:	`(x_2 >>= 1)`	(x_2)	()
23:	`(x_2 *= 4)`	(x_2)	()
24:	`(x_2 += a)`	(a x_2)	()
25:	`((mem x_2 4) <- newaval)`	(newaval x_2)	()
26:	`(_set <- 1)`	()	()
27:	`(nexti <- i)`	(i)	()
28:	`(nexti += 2)`	(nexti)	()
29:	`(ecx <- a)`	(a)	()
30:	`(edx <- nexti)`	(nexti)	()
31:	`(eax <- y)`	(y)	()
32:	`(goto :inc)`	()	()
33:	`:L_2`	()	()
34:	`(eax <- 1)`	()	()
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	()	(ecx)
2:	`(a <- ecx)`	(ecx)	(edx)
3:	`(i <- edx)`	(edx)	(eax)
4:	`(y <- eax)`	(eax)	(a)
5:	`(len <- (mem a 0))`	(a)	(len)
6:	`(len <<= 1)`	(len)	(len)
7:	`(len += 1)`	(len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen)	(altlen)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen)	()
12:	`:L_1`	()	(i)
13:	`(x_1 <- i)`	(i)	(x_1)
14:	`(x_1 >>= 1)`	(x_1)	(x_1)
15:	`(x_1 *= 4)`	(x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1)	(x_1)
17:	`(aval <- (mem x_1 4))`	(x_1)	(aval)
18:	`(newaval <- aval)`	(aval)	(newaval y)
19:	`(newaval += y)`	(newaval y)	(newaval)
20:	`(newaval -= 1)`	(newaval)	(i)
21:	`(x_2 <- i)`	(i)	(x_2)
22:	`(x_2 >>= 1)`	(x_2)	(x_2)
23:	`(x_2 *= 4)`	(x_2)	(a x_2)
24:	`(x_2 += a)`	(a x_2)	(newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(newaval x_2)	()
26:	`(_set <- 1)`	()	(i)
27:	`(nexti <- i)`	(i)	(nexti)
28:	`(nexti += 2)`	(nexti)	(a)
29:	`(ecx <- a)`	(a)	(nexti)
30:	`(edx <- nexti)`	(nexti)	(y)
31:	`(eax <- y)`	(y)	()
32:	`(goto :inc)`	()	()
33:	`:L_2`	()	()
34:	`(eax <- 1)`	()	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(ecx)	(ecx)
2:	`(a <- ecx)`	(ecx edx)	(edx)
3:	`(i <- edx)`	(eax edx)	(eax)
4:	`(y <- eax)`	(a eax)	(a)
5:	`(len <- (mem a 0))`	(a)	(len)
6:	`(len <<= 1)`	(len)	(len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen)	(altlen)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen)	()
12:	`:L_1`	(i)	(i)
13:	`(x_1 <- i)`	(i)	(x_1)
14:	`(x_1 >>= 1)`	(x_1)	(x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1)	(x_1)
17:	`(aval <- (mem x_1 4))`	(x_1)	(aval)
18:	`(newaval <- aval)`	(aval y)	(newaval y)
19:	`(newaval += y)`	(newaval y)	(newaval)
20:	`(newaval -= 1)`	(i newaval)	(i)
21:	`(x_2 <- i)`	(i)	(x_2)
22:	`(x_2 >>= 1)`	(x_2)	(x_2)
23:	`(x_2 *= 4)`	(a x_2)	(a x_2)
24:	`(x_2 += a)`	(a newaval x_2)	(newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(newaval x_2)	()
26:	`(_set <- 1)`	(i)	(i)
27:	`(nexti <- i)`	(i)	(nexti)
28:	`(nexti += 2)`	(a nexti)	(a)
29:	`(ecx <- a)`	(a nexti)	(nexti)
30:	`(edx <- nexti)`	(nexti y)	(y)
31:	`(eax <- y)`	(y)	()
32:	`(goto :inc)`	()	()
33:	`:L_2`	()	()
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(ecx)	(ecx edx)
2:	`(a <- ecx)`	(ecx edx)	(eax edx)
3:	`(i <- edx)`	(eax edx)	(a eax)
4:	`(y <- eax)`	(a eax)	(a)
5:	`(len <- (mem a 0))`	(a)	(len)
6:	`(len <<= 1)`	(len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen)	(altlen)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen)	(i)
12:	`:L_1`	(i)	(i)
13:	`(x_1 <- i)`	(i)	(x_1)
14:	`(x_1 >>= 1)`	(x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1)	(x_1)
17:	`(aval <- (mem x_1 4))`	(x_1)	(aval y)
18:	`(newaval <- aval)`	(aval y)	(newaval y)
19:	`(newaval += y)`	(newaval y)	(i newaval)
20:	`(newaval -= 1)`	(i newaval)	(i)
21:	`(x_2 <- i)`	(i)	(x_2)
22:	`(x_2 >>= 1)`	(x_2)	(a x_2)
23:	`(x_2 *= 4)`	(a x_2)	(a newaval x_2)
24:	`(x_2 += a)`	(a newaval x_2)	(newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(newaval x_2)	(i)
26:	`(_set <- 1)`	(i)	(i)
27:	`(nexti <- i)`	(i)	(a nexti)
28:	`(nexti += 2)`	(a nexti)	(a nexti)
29:	`(ecx <- a)`	(a nexti)	(nexti y)
30:	`(edx <- nexti)`	(nexti y)	(y)
31:	`(eax <- y)`	(y)	()
32:	`(goto :inc)`	()	(ecx)
33:	`:L_2`	()	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(ecx edx)	(ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax)
4:	`(y <- eax)`	(a eax)	(a)
5:	`(len <- (mem a 0))`	(a)	(len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen)	(altlen)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen i)	(i)
12:	`:L_1`	(i)	(i)
13:	`(x_1 <- i)`	(i)	(x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1)	(x_1)
17:	`(aval <- (mem x_1 4))`	(x_1 y)	(aval y)
18:	`(newaval <- aval)`	(aval y)	(newaval y)
19:	`(newaval += y)`	(i newaval y)	(i newaval)
20:	`(newaval -= 1)`	(i newaval)	(i)
21:	`(x_2 <- i)`	(i)	(x_2)
22:	`(x_2 >>= 1)`	(a x_2)	(a x_2)
23:	`(x_2 *= 4)`	(a newaval x_2)	(a newaval x_2)
24:	`(x_2 += a)`	(a newaval x_2)	(newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(i newaval x_2)	(i)
26:	`(_set <- 1)`	(i)	(i)
27:	`(nexti <- i)`	(a i)	(a nexti)
28:	`(nexti += 2)`	(a nexti)	(a nexti)
29:	`(ecx <- a)`	(a nexti y)	(nexti y)
30:	`(edx <- nexti)`	(nexti y)	(y)
31:	`(eax <- y)`	(y)	()
32:	`(goto :inc)`	(ecx)	(ecx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax)
4:	`(y <- eax)`	(a eax)	(a)
5:	`(len <- (mem a 0))`	(a)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen)	(altlen i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen i)	(ebx edi esi i)
12:	`:L_1`	(i)	(i)
13:	`(x_1 <- i)`	(i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1)	(x_1 y)
17:	`(aval <- (mem x_1 4))`	(x_1 y)	(aval y)
18:	`(newaval <- aval)`	(aval y)	(i newaval y)
19:	`(newaval += y)`	(i newaval y)	(i newaval)
20:	`(newaval -= 1)`	(i newaval)	(i)
21:	`(x_2 <- i)`	(i)	(a x_2)
22:	`(x_2 >>= 1)`	(a x_2)	(a newaval x_2)
23:	`(x_2 *= 4)`	(a newaval x_2)	(a newaval x_2)
24:	`(x_2 += a)`	(a newaval x_2)	(i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(i newaval x_2)	(i)
26:	`(_set <- 1)`	(i)	(a i)
27:	`(nexti <- i)`	(a i)	(a nexti)
28:	`(nexti += 2)`	(a nexti)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(nexti y)
30:	`(edx <- nexti)`	(nexti y)	(y)
31:	`(eax <- y)`	(y)	(ecx)
32:	`(goto :inc)`	(ecx)	(ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax)
4:	`(y <- eax)`	(a eax)	(a)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen)
10:	`(altlen += 1)`	(altlen i)	(altlen i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen ebx edi esi i)	(ebx edi esi i)
12:	`:L_1`	(i)	(i)
13:	`(x_1 <- i)`	(a i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1)
16:	`(x_1 += a)`	(a x_1 y)	(x_1 y)
17:	`(aval <- (mem x_1 4))`	(x_1 y)	(aval y)
18:	`(newaval <- aval)`	(aval i y)	(i newaval y)
19:	`(newaval += y)`	(i newaval y)	(i newaval)
20:	`(newaval -= 1)`	(i newaval)	(i)
21:	`(x_2 <- i)`	(a i)	(a x_2)
22:	`(x_2 >>= 1)`	(a newaval x_2)	(a newaval x_2)
23:	`(x_2 *= 4)`	(a newaval x_2)	(a newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(i newaval x_2)	(i)
26:	`(_set <- 1)`	(a i)	(a i)
27:	`(nexti <- i)`	(a i)	(a nexti)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(nexti y)
30:	`(edx <- nexti)`	(nexti y)	(y)
31:	`(eax <- y)`	(ecx y)	(ecx)
32:	`(goto :inc)`	(ecx edx)	(ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax)
4:	`(y <- eax)`	(a eax)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen)	(altlen i)
10:	`(altlen += 1)`	(altlen i)	(altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen ebx edi esi i)	(ebx edi esi i)
12:	`:L_1`	(i)	(a i)
13:	`(x_1 <- i)`	(a i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1)	(a x_1 y)
16:	`(x_1 += a)`	(a x_1 y)	(x_1 y)
17:	`(aval <- (mem x_1 4))`	(x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(aval i y)	(i newaval y)
19:	`(newaval += y)`	(i newaval y)	(i newaval)
20:	`(newaval -= 1)`	(i newaval)	(a i)
21:	`(x_2 <- i)`	(a i)	(a newaval x_2)
22:	`(x_2 >>= 1)`	(a newaval x_2)	(a newaval x_2)
23:	`(x_2 *= 4)`	(a newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(i newaval x_2)	(a i)
26:	`(_set <- 1)`	(a i)	(a i)
27:	`(nexti <- i)`	(a i)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(nexti y)
30:	`(edx <- nexti)`	(nexti y)	(ecx y)
31:	`(eax <- y)`	(ecx y)	(ecx edx)
32:	`(goto :inc)`	(ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen)
9:	`(altlen += altlen)`	(altlen i)	(altlen i)
10:	`(altlen += 1)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen ebx edi esi i)	(ebx edi esi i)
12:	`:L_1`	(a i)	(a i)
13:	`(x_1 <- i)`	(a i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1)
15:	`(x_1 *= 4)`	(a x_1 y)	(a x_1 y)
16:	`(x_1 += a)`	(a x_1 y)	(x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(aval i y)	(i newaval y)
19:	`(newaval += y)`	(i newaval y)	(i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i)
21:	`(x_2 <- i)`	(a i newaval)	(a newaval x_2)
22:	`(x_2 >>= 1)`	(a newaval x_2)	(a newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2)	(a i)
26:	`(_set <- 1)`	(a i)	(a i)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx y)
31:	`(eax <- y)`	(ecx edx y)	(ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen i)
9:	`(altlen += altlen)`	(altlen i)	(altlen ebx edi esi i)
10:	`(altlen += 1)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i)	(a i)
13:	`(x_1 <- i)`	(a i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1)	(a x_1 y)
15:	`(x_1 *= 4)`	(a x_1 y)	(a x_1 y)
16:	`(x_1 += a)`	(a x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(aval i y)	(i newaval y)
19:	`(newaval += y)`	(i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a newaval x_2)
22:	`(x_2 >>= 1)`	(a newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(a i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2)	(a i)
26:	`(_set <- 1)`	(a i)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen i)
9:	`(altlen += altlen)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
10:	`(altlen += 1)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i)	(a i)
13:	`(x_1 <- i)`	(a i)	(a x_1)
14:	`(x_1 >>= 1)`	(a x_1 y)	(a x_1 y)
15:	`(x_1 *= 4)`	(a x_1 y)	(a x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(aval i y)	(i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(a i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2)	(a i)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(i len)	(altlen ebx edi esi i)
9:	`(altlen += altlen)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
10:	`(altlen += 1)`	(altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i)	(a i)
13:	`(x_1 <- i)`	(a i)	(a x_1 y)
14:	`(x_1 >>= 1)`	(a x_1 y)	(a x_1 y)
15:	`(x_1 *= 4)`	(a x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(a i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(i len)
8:	`(altlen <- i < len)`	(ebx edi esi i len)	(altlen ebx edi esi i)
9:	`(altlen += altlen)`	(altlen ebx edi esi i)	(altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i)	(a i)
13:	`(x_1 <- i)`	(a i y)	(a x_1 y)
14:	`(x_1 >>= 1)`	(a x_1 y)	(a x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(a i newaval x_2)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(i len)	(ebx edi esi i len)
8:	`(altlen <- i < len)`	(ebx edi esi i len)	(altlen ebx edi esi i)
9:	`(altlen += altlen)`	(altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a x_1 y)
14:	`(x_1 >>= 1)`	(a x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(i len)
7:	`(len += 1)`	(ebx edi esi i len)	(ebx edi esi i len)
8:	`(altlen <- i < len)`	(ebx edi esi i len)	(altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(i len)	(ebx edi esi i len)
7:	`(len += 1)`	(ebx edi esi i len)	(ebx edi esi i len)
8:	`(altlen <- i < len)`	(ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(i len)
6:	`(len <<= 1)`	(ebx edi esi i len)	(ebx edi esi i len)
7:	`(len += 1)`	(ebx edi esi i len)	(ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a i)	(ebx edi esi i len)
6:	`(len <<= 1)`	(ebx edi esi i len)	(ebx edi esi i len)
7:	`(len += 1)`	(ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(ebx edi esi i len)
6:	`(len <<= 1)`	(ebx edi esi i len)	(ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(ebx edi esi i len)
6:	`(len <<= 1)`	(ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax edx)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax edx)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ecx edx)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ecx edx)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ebx ecx edi edx esi)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len)	(a ebx edi esi i len y)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ebx ecx edi edx esi)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len)
6:	`(len <<= 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ebx ecx edi edx esi)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i)	(a ebx edi esi i len y)
6:	`(len <<= 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ebx ecx edi edx esi)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i)
5:	`(len <- (mem a 0))`	(a ebx edi esi i y)	(a ebx edi esi i len y)
6:	`(len <<= 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

		in	out
1:	`:inc`	(eax ecx edx)	(eax ebx ecx edi edx esi)
2:	`(a <- ecx)`	(eax ebx ecx edi edx esi)	(a eax ebx edi edx esi)
3:	`(i <- edx)`	(a eax ebx edi edx esi)	(a eax ebx edi esi i)
4:	`(y <- eax)`	(a eax ebx edi esi i)	(a ebx edi esi i y)
5:	`(len <- (mem a 0))`	(a ebx edi esi i y)	(a ebx edi esi i len y)
6:	`(len <<= 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
7:	`(len += 1)`	(a ebx edi esi i len y)	(a ebx edi esi i len y)
8:	`(altlen <- i < len)`	(a ebx edi esi i len y)	(a altlen ebx edi esi i y)
9:	`(altlen += altlen)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
10:	`(altlen += 1)`	(a altlen ebx edi esi i y)	(a altlen ebx edi esi i y)
11:	`(cjump altlen = 3 :L_1 :L_2)`	(a altlen ebx edi esi i y)	(a ebx edi esi i y)
12:	`:L_1`	(a i y)	(a i y)
13:	`(x_1 <- i)`	(a i y)	(a i x_1 y)
14:	`(x_1 >>= 1)`	(a i x_1 y)	(a i x_1 y)
15:	`(x_1 *= 4)`	(a i x_1 y)	(a i x_1 y)
16:	`(x_1 += a)`	(a i x_1 y)	(a i x_1 y)
17:	`(aval <- (mem x_1 4))`	(a i x_1 y)	(a aval i y)
18:	`(newaval <- aval)`	(a aval i y)	(a i newaval y)
19:	`(newaval += y)`	(a i newaval y)	(a i newaval y)
20:	`(newaval -= 1)`	(a i newaval y)	(a i newaval y)
21:	`(x_2 <- i)`	(a i newaval y)	(a i newaval x_2 y)
22:	`(x_2 >>= 1)`	(a i newaval x_2 y)	(a i newaval x_2 y)
23:	`(x_2 *= 4)`	(a i newaval x_2 y)	(a i newaval x_2 y)
24:	`(x_2 += a)`	(a i newaval x_2 y)	(a i newaval x_2 y)
25:	`((mem x_2 4) <- newaval)`	(a i newaval x_2 y)	(a i y)
26:	`(_set <- 1)`	(a i y)	(a i y)
27:	`(nexti <- i)`	(a i y)	(a nexti y)
28:	`(nexti += 2)`	(a nexti y)	(a nexti y)
29:	`(ecx <- a)`	(a nexti y)	(ecx nexti y)
30:	`(edx <- nexti)`	(ecx nexti y)	(ecx edx y)
31:	`(eax <- y)`	(ecx edx y)	(eax ecx edx)
32:	`(goto :inc)`	(eax ecx edx)	(eax ecx edx)
33:	`:L_2`	(ebx edi esi)	(ebx edi esi)
34:	`(eax <- 1)`	(ebx edi esi)	(ebx edi esi)
35:	`(return)`	(ebx edi esi)	()

Line 26, `_set` is not in the out set, so that's dead code; we can drop it

```
:inc                              :inc
(a <- ecx)                        (a <- ecx)
(i <- edx)                        (i <- edx)
(y <- eax)                        (y <- eax)
(len <- (mem a 0))                (len <- (mem a 0))
(len <<= 1)                       (len <<= 1)
(len += 1)                        (len += 1)
(altlen <- i < len)               (altlen <- i < len)
(altlen += altlen)                (altlen += altlen)
(altlen += 1)                     (altlen += 1)
(cjump altlen = 3 :L_1 :L_2)      (cjump altlen = 3 :L_1 :L_2)
:L_1                              :L_1
(x_1 <- i)                        (x_1 <- i)
(x_1 >>= 1)                       (x_1 >>= 1)
(x_1 *= 4)                        (x_1 *= 4)
(x_1 += a)                        (x_1 += a)
(aval <- (mem x_1 4))             (aval <- (mem x_1 4))
(newaval <- aval)                 (newaval <- aval)
(newaval += y)                    (newaval += y)
(newaval -= 1)                    (newaval -= 1)
(x_2 <- i)                        (x_2 <- i)
(x_2 >>= 1)                       (x_2 >>= 1)
(x_2 *= 4)                        (x_2 *= 4)
(x_2 += a)                        (x_2 += a)
((mem x_2 4) <- newaval)          ((mem x_2 4) <- newaval)
(_set <- 1)
(nexti <- i)                      (nexti <- i)
(nexti += 2)                      (nexti += 2)
(ecx <- a)                        (ecx <- a)
(edx <- nexti)                    (edx <- nexti)
(eax <- y)                        (eax <- y)
(goto :inc)                       (goto :inc)
:L_2                              :L_2
(eax <- 1)                        (eax <- 1)
(return)                          (return)
```

To do more with this program, we need a more sophisticated analysis.

For each statement, collect a set of expressions that have been computed and the variables they belong to.

This is something similar to **value numbering** but adapted to our setting; typically this is done with three address code, but we have two address code

This is a forwards data flow (opposite from liveness), so the in set is computed as the intersection of the outs of the predecessors.

The out set is the in set, but with adjustments to the sets made by this instruction; memory references are ignored, since don't know when two memory addresses are the same

in,out : inst# → (x → expr)
in(1) = ∅
in(s) = ∩{out(s') | s' ∈ prec(s)}
out(s) = (match (inst s)
[`((mem ,x ,y) <- ,s) in(s)]
[`(,x <- (mem ,y ,z)) in(s) \ x]
[`(,x <- ,s) in(s) ⊕ {x → s}]
[`(,x ,op= ,y) in(s) ⊕ {x → (in(s)(x) op y)}]
[else in(s)])

where ⊕ replaces a mapping and \ removes a mapping completely

Here are the first few iterations of the computation for our program (watch what happens with **`x_1`** specifically, lines 13–16)

```
 1: :inc                          ()  ()
 2: (a <- ecx)                    ()  ()
 3: (i <- edx)                    ()  ()
 4: (y <- eax)                    ()  ()
 5: (len <- (mem a 0))            ()  ()
 6: (len <<= 1)                   ()  ()
 7: (len += 1)                    ()  ()
 8: (altlen <- i < len)           ()  ()
 9: (altlen += altlen)            ()  ()
10: (altlen += 1)                 ()  ()
11: (cjump altlen = 3 :L_1 :L_2)  ()  ()
12: :L_1                          ()  ()
13: (x_1 <- i)                    ()  ()
14: (x_1 >>= 1)                   ()  ()
15: (x_1 *= 4)                    ()  ()
16: (x_1 += a)                    ()  ()
17: (aval <- (mem x_1 4))         ()  ()
18: (newaval <- aval)             ()  ()
19: (newaval += y)                ()  ()
20: (newaval -= 1)                ()  ()
21: (x_2 <- i)                    ()  ()
22: (x_2 >>= 1)                   ()  ()
23: (x_2 *= 4)                    ()  ()
24: (x_2 += a)                    ()  ()
25: ((mem x_2 4) <- newaval)      ()  ()
26: (nexti <- i)                  ()  ()
27: (nexti += 2)                  ()  ()
28: (ecx <- a)                    ()  ()
29: (edx <- nexti)                ()  ()
30: (eax <- y)                    ()  ()
31: (goto :inc)                   ()  ()
32: :L_2                          ()  ()
33: (eax <- 1)                    ()  ()
34: (return)                      ()  ()
```

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	()	({edx i})
4:	`(y <- eax)`	()	({eax y})
5:	`(len <- (mem a 0))`	()	()
6:	`(len <<= 1)`	()	()
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	()	()
15:	`(x_1 *= 4)`	()	()
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	()	()
20:	`(newaval -= 1)`	()	()
21:	`(x_2 <- i)`	()	({i x_2})
22:	`(x_2 >>= 1)`	()	()
23:	`(x_2 *= 4)`	()	()
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	()	()
28:	`(ecx <- a)`	()	({a ecx})
29:	`(edx <- nexti)`	()	({nexti edx})
30:	`(eax <- y)`	()	({y eax})
31:	`(goto :inc)`	()	()
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	()	()

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i})
4:	`(y <- eax)`	({edx i})	({eax y})
5:	`(len <- (mem a 0))`	({eax y})	()
6:	`(len <<= 1)`	()	()
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	()
15:	`(x_1 *= 4)`	()	()
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	()
20:	`(newaval -= 1)`	()	()
21:	`(x_2 <- i)`	()	({i x_2})
22:	`(x_2 >>= 1)`	({i x_2})	()
23:	`(x_2 *= 4)`	()	()
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	()
28:	`(ecx <- a)`	()	({a ecx})
29:	`(edx <- nexti)`	({a ecx})	({nexti edx})
30:	`(eax <- y)`	({nexti edx})	({y eax})
31:	`(goto :inc)`	({y eax})	()
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	()

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({edx i})	({eax y} {edx i})
5:	`(len <- (mem a 0))`	({eax y})	({eax y})
6:	`(len <<= 1)`	()	()
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	()	()
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	()	()
21:	`(x_2 <- i)`	()	({i x_2})
22:	`(x_2 >>= 1)`	({i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	()	()
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	()	({a ecx})
29:	`(edx <- nexti)`	({a ecx})	({nexti edx} {a ecx})
30:	`(eax <- y)`	({nexti edx})	({y eax} {nexti edx})
31:	`(goto :inc)`	({y eax})	({y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {edx i})	({eax y})
6:	`(len <<= 1)`	({eax y})	()
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	()
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	()
21:	`(x_2 <- i)`	()	({i x_2})
22:	`(x_2 >>= 1)`	({i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	()
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx})
29:	`(edx <- nexti)`	({a ecx})	({nexti edx} {a ecx})
30:	`(eax <- y)`	({a ecx} {nexti edx})	({y eax} {nexti edx})
31:	`(goto :inc)`	({nexti edx} {y eax})	({y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {ecx a} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {edx i})	({eax y} {edx i})
6:	`(len <<= 1)`	({eax y})	({eax y})
7:	`(len += 1)`	()	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	({((i >> 1) * 4) x_1})
16:	`(x_1 += a)`	()	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	({((aval + y) - 1) newaval})
21:	`(x_2 <- i)`	()	({i x_2})
22:	`(x_2 >>= 1)`	({i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	({((i >> 1) * 4) x_2})
24:	`(x_2 += a)`	()	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx} {(i + 2) nexti})
29:	`(edx <- nexti)`	({a ecx})	({nexti edx} {a ecx})
30:	`(eax <- y)`	({a ecx} {nexti edx})	({y eax} {a ecx} {nexti edx})
31:	`(goto :inc)`	({nexti edx} {y eax})	({nexti edx} {y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {ecx a} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {ecx a} {edx i})	({eax y} {edx i})
6:	`(len <<= 1)`	({eax y} {edx i})	({eax y})
7:	`(len += 1)`	({eax y})	()
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	({((i >> 1) * 4) x_1})
16:	`(x_1 += a)`	({((i >> 1) * 4) x_1})	()
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	({((aval + y) - 1) newaval})
21:	`(x_2 <- i)`	({((aval + y) - 1) newaval})	({i x_2})
22:	`(x_2 >>= 1)`	({i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	({((i >> 1) * 4) x_2})
24:	`(x_2 += a)`	({((i >> 1) * 4) x_2})	()
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx} {(i + 2) nexti})
29:	`(edx <- nexti)`	({(i + 2) nexti} {a ecx})	({nexti edx} {a ecx})
30:	`(eax <- y)`	({a ecx} {nexti edx})	({y eax} {a ecx} {nexti edx})
31:	`(goto :inc)`	({a ecx} {nexti edx} {y eax})	({nexti edx} {y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {ecx a} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {ecx a} {edx i})	({eax y} {ecx a} {edx i})
6:	`(len <<= 1)`	({eax y} {edx i})	({eax y} {edx i})
7:	`(len += 1)`	({eax y})	({eax y})
8:	`(altlen <- i < len)`	()	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	({((i >> 1) * 4) x_1})
16:	`(x_1 += a)`	({((i >> 1) * 4) x_1})	({(((i >> 1) * 4) + a) x_1})
17:	`(aval <- (mem x_1 4))`	()	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	({((aval + y) - 1) newaval})
21:	`(x_2 <- i)`	({((aval + y) - 1) newaval})	({i x_2} {((aval + y) - 1) newaval})
22:	`(x_2 >>= 1)`	({i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	({((i >> 1) * 4) x_2})
24:	`(x_2 += a)`	({((i >> 1) * 4) x_2})	({(((i >> 1) * 4) + a) x_2})
25:	`((mem x_2 4) <- newaval)`	()	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx} {(i + 2) nexti})
29:	`(edx <- nexti)`	({(i + 2) nexti} {a ecx})	({nexti edx} {(i + 2) nexti} {a ecx})
30:	`(eax <- y)`	({a ecx} {nexti edx})	({y eax} {a ecx} {nexti edx})
31:	`(goto :inc)`	({a ecx} {nexti edx} {y eax})	({a ecx} {nexti edx} {y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {ecx a} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {ecx a} {edx i})	({eax y} {ecx a} {edx i})
6:	`(len <<= 1)`	({eax y} {ecx a} {edx i})	({eax y} {edx i})
7:	`(len += 1)`	({eax y} {edx i})	({eax y})
8:	`(altlen <- i < len)`	({eax y})	()
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	({((i >> 1) * 4) x_1})
16:	`(x_1 += a)`	({((i >> 1) * 4) x_1})	({(((i >> 1) * 4) + a) x_1})
17:	`(aval <- (mem x_1 4))`	({(((i >> 1) * 4) + a) x_1})	()
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	({((aval + y) - 1) newaval})
21:	`(x_2 <- i)`	({((aval + y) - 1) newaval})	({i x_2} {((aval + y) - 1) newaval})
22:	`(x_2 >>= 1)`	({((aval + y) - 1) newaval} {i x_2})	({(i >> 1) x_2})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	({((i >> 1) * 4) x_2})
24:	`(x_2 += a)`	({((i >> 1) * 4) x_2})	({(((i >> 1) * 4) + a) x_2})
25:	`((mem x_2 4) <- newaval)`	({(((i >> 1) * 4) + a) x_2})	()
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx} {(i + 2) nexti})
29:	`(edx <- nexti)`	({(i + 2) nexti} {a ecx})	({nexti edx} {(i + 2) nexti} {a ecx})
30:	`(eax <- y)`	({(i + 2) nexti} {a ecx} {nexti edx})	({y eax} {a ecx} {nexti edx})
31:	`(goto :inc)`	({a ecx} {nexti edx} {y eax})	({a ecx} {nexti edx} {y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

1:	`:inc`	()	()
2:	`(a <- ecx)`	()	({ecx a})
3:	`(i <- edx)`	({ecx a})	({edx i} {ecx a})
4:	`(y <- eax)`	({ecx a} {edx i})	({eax y} {ecx a} {edx i})
5:	`(len <- (mem a 0))`	({eax y} {ecx a} {edx i})	({eax y} {ecx a} {edx i})
6:	`(len <<= 1)`	({eax y} {ecx a} {edx i})	({eax y} {ecx a} {edx i})
7:	`(len += 1)`	({eax y} {edx i})	({eax y} {edx i})
8:	`(altlen <- i < len)`	({eax y})	({eax y})
9:	`(altlen += altlen)`	()	()
10:	`(altlen += 1)`	()	()
11:	`(cjump altlen = 3 :L_1 :L_2)`	()	()
12:	`:L_1`	()	()
13:	`(x_1 <- i)`	()	({i x_1})
14:	`(x_1 >>= 1)`	({i x_1})	({(i >> 1) x_1})
15:	`(x_1 *= 4)`	({(i >> 1) x_1})	({((i >> 1) * 4) x_1})
16:	`(x_1 += a)`	({((i >> 1) * 4) x_1})	({(((i >> 1) * 4) + a) x_1})
17:	`(aval <- (mem x_1 4))`	({(((i >> 1) * 4) + a) x_1})	({(((i >> 1) * 4) + a) x_1})
18:	`(newaval <- aval)`	()	({aval newaval})
19:	`(newaval += y)`	({aval newaval})	({(aval + y) newaval})
20:	`(newaval -= 1)`	({(aval + y) newaval})	({((aval + y) - 1) newaval})
21:	`(x_2 <- i)`	({((aval + y) - 1) newaval})	({i x_2} {((aval + y) - 1) newaval})
22:	`(x_2 >>= 1)`	({((aval + y) - 1) newaval} {i x_2})	({(i >> 1) x_2} {((aval + y) - 1) newaval})
23:	`(x_2 *= 4)`	({(i >> 1) x_2})	({((i >> 1) * 4) x_2})
24:	`(x_2 += a)`	({((i >> 1) * 4) x_2})	({(((i >> 1) * 4) + a) x_2})
25:	`((mem x_2 4) <- newaval)`	({(((i >> 1) * 4) + a) x_2})	({(((i >> 1) * 4) + a) x_2})
26:	`(nexti <- i)`	()	({i nexti})
27:	`(nexti += 2)`	({i nexti})	({(i + 2) nexti})
28:	`(ecx <- a)`	({(i + 2) nexti})	({a ecx} {(i + 2) nexti})
29:	`(edx <- nexti)`	({(i + 2) nexti} {a ecx})	({nexti edx} {(i + 2) nexti} {a ecx})
30:	`(eax <- y)`	({(i + 2) nexti} {a ecx} {nexti edx})	({y eax} {(i + 2) nexti} {a ecx} {nexti edx})
31:	`(goto :inc)`	({a ecx} {nexti edx} {y eax})	({a ecx} {nexti edx} {y eax})
32:	`:L_2`	()	()
33:	`(eax <- 1)`	()	({1 eax})
34:	`(return)`	({1 eax})	({1 eax})

Here are the complete results for the available expressions, in tabular form

	newaval	nexti	x_1	x_2
:inc				
(a <- ecx)				
(i <- edx)				
(y <- eax)				
(len <- (mem a 0))				
(len <<= 1)				
(len += 1)				
(altlen <- i < len)				
(altlen += altlen)				
(altlen += 1)				
(cjump altlen = 3 :L_1 :L_2)				
:L_1				
(x_1 <- i)			i	
(x_1 >>= 1)			(i >> 1)	
(x_1 *= 4)			((i >> 1) * 4)	
(x_1 += a)			(((i >> 1) * 4) + a)	
(aval <- (mem x_1 4))			(((i >> 1) * 4) + a)	
(newaval <- aval)	aval		(((i >> 1) * 4) + a)	
(newaval += y)	(aval + y)		(((i >> 1) * 4) + a)	
(newaval -= 1)	((aval + y) - 1)		(((i >> 1) * 4) + a)	
(x_2 <- i)	((aval + y) - 1)		(((i >> 1) * 4) + a)	i
(x_2 >>= 1)	((aval + y) - 1)		(((i >> 1) * 4) + a)	(i >> 1)
(x_2 *= 4)	((aval + y) - 1)		(((i >> 1) * 4) + a)	((i >> 1) * 4)
(x_2 += a)	((aval + y) - 1)		(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
((mem x_2 4) <- newaval)	((aval + y) - 1)		(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(nexti <- i)	((aval + y) - 1)	i	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(nexti += 2)	((aval + y) - 1)	(i + 2)	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(ecx <- a)	((aval + y) - 1)	(i + 2)	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(edx <- nexti)	((aval + y) - 1)	(i + 2)	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(eax <- y)	((aval + y) - 1)	(i + 2)	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
(goto :inc)	((aval + y) - 1)	(i + 2)	(((i >> 1) * 4) + a)	(((i >> 1) * 4) + a)
:L_2				
(eax <- 1)				
(return)				

We can use this analysis to rewrite a program to use the result of an earlier computation, instead of a new computation

More precisely, if we have `(x op= s)` and the expression for `x` in the out set of that instruction is the same as the expression for some other variable `y`, then we can rewrite the instruction to `(x <- y)`

```
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(newaval += y)
(newaval -= 1)
(x_2 <- i)
(x_2 >>= 1)
(x_2 *= 4)
(x_2 += a)
((mem x_2 4) <- newaval)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)
```

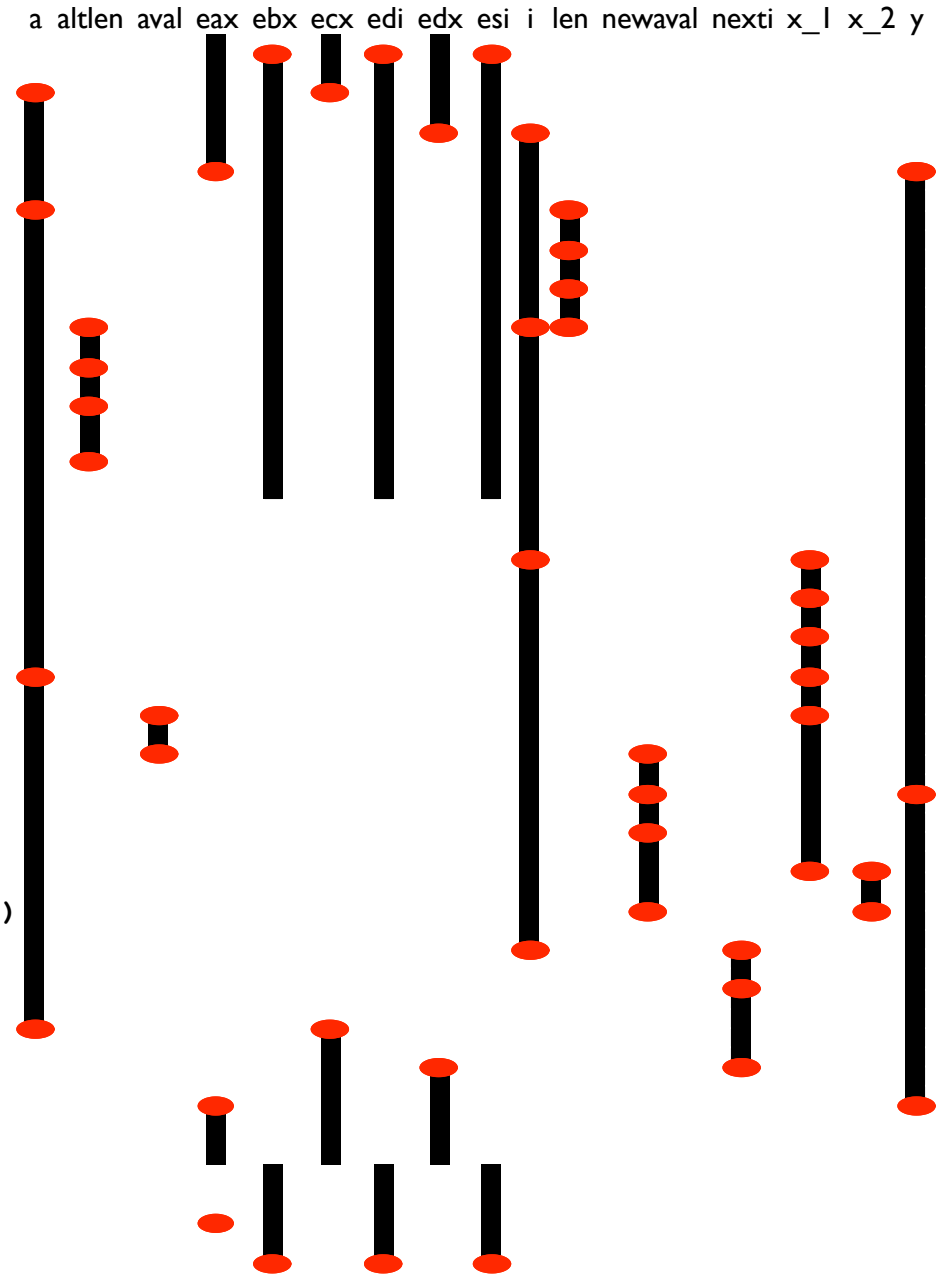
```
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(newaval += y)
(newaval -= 1)
(x_2 <- i)
(x_2 >>= 1)
(x_2 *= 4)
(x_2 <- x_1)
((mem x_2 4) <- newaval)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)
```

Now we can use dead code elimination to drop the earlier computations

```
:inc                              :inc
(a <- ecx)                        (a <- ecx)
(i <- edx)                        (i <- edx)
(y <- eax)                        (y <- eax)
(len <- (mem a 0))                (len <- (mem a 0))
(len <<= 1)                       (len <<= 1)
(len += 1)                        (len += 1)
(altlen <- i < len)               (altlen <- i < len)
(altlen += altlen)                (altlen += altlen)
(altlen += 1)                     (altlen += 1)
(cjump altlen = 3 :L_1 :L_2)      (cjump altlen = 3 :L_1 :L_2)
:L_1                              :L_1
(x_1 <- i)                        (x_1 <- i)
(x_1 >>= 1)                       (x_1 >>= 1)
(x_1 *= 4)                        (x_1 *= 4)
(x_1 += a)                        (x_1 += a)
(aval <- (mem x_1 4))             (aval <- (mem x_1 4))
(newaval <- aval)                 (newaval <- aval)
(newaval += y)                    (newaval += y)
(newaval -= 1)                    (newaval -= 1)
(x_2 <- i)
(x_2 >>= 1)
(x_2 *= 4)
(x_2 <- x_1)                      (x_2 <- x_1)
((mem x_2 4) <- newaval)          ((mem x_2 4) <- newaval)
(nexti <- i)                      (nexti <- i)
(nexti += 2)                      (nexti += 2)
(ecx <- a)                        (ecx <- a)
(edx <- nexti)                    (edx <- nexti)
(eax <- y)                        (eax <- y)
(goto :inc)                       (goto :inc)
:L_2                              :L_2
(eax <- 1)                        (eax <- 1)
(return)                          (return)
```

How does register allocation fare here?

```
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3
       :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(newaval += y)
(newaval -= 1)
(x_2 <- x_1)
((mem x_2 4) <- newaval)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)
```



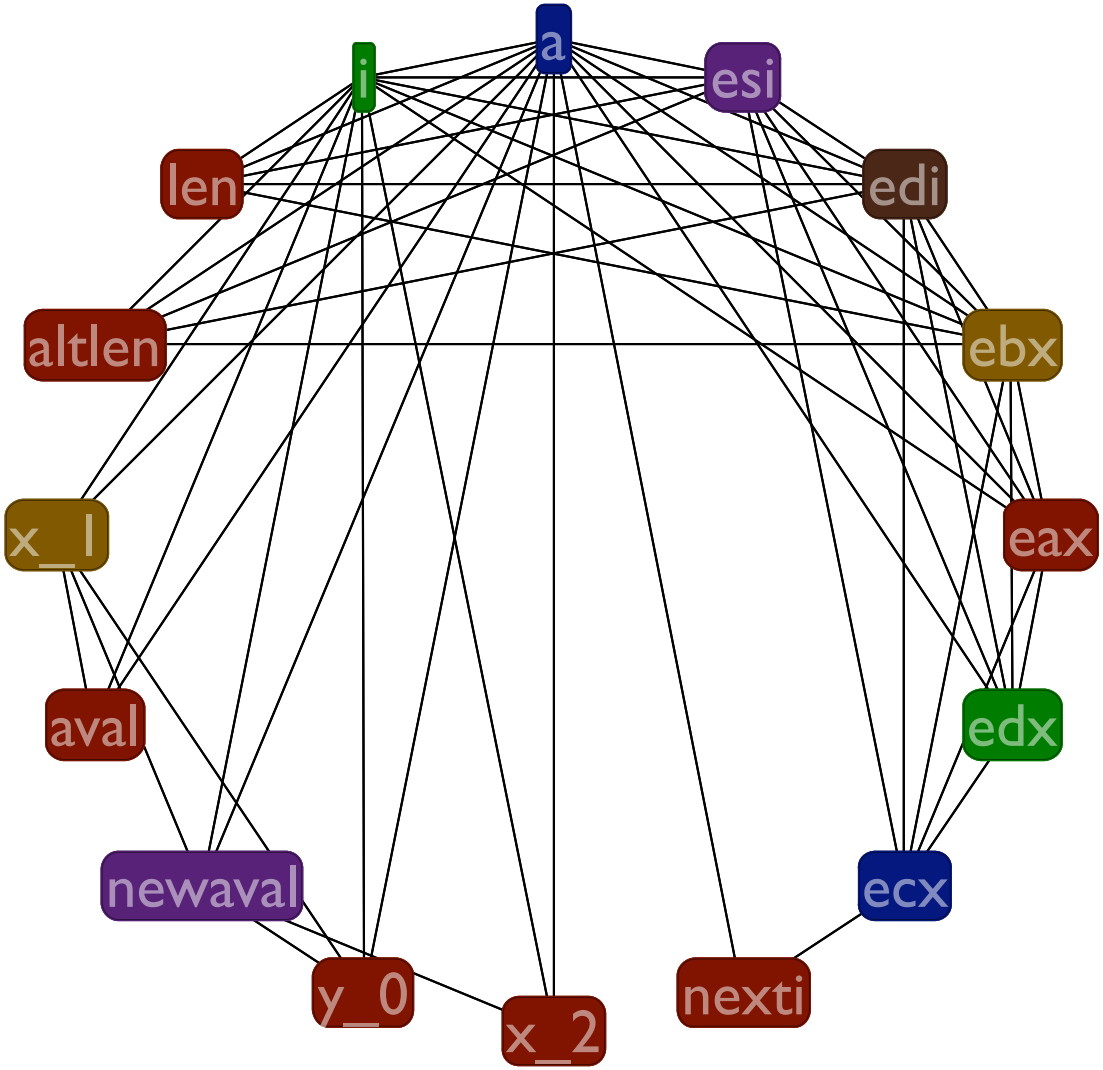
There are 7 active variables at once; so we have to spill; spilling `y` we get this program

```
:inc
(a <- ecx)
(i <- edx)
(y <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)

(newaval += y)
(newaval -= 1)
(x_2 <- x_1)
((mem x_2 4) <- newaval)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- y)
(goto :inc)
:L_2
(eax <- 1)
(return)
```

```
:inc
(a <- ecx)
(i <- edx)
((mem ebp -4) <- eax)
(len <- (mem a 0))
(len <<= 1)
(len += 1)
(altlen <- i < len)
(altlen += altlen)
(altlen += 1)
(cjump altlen = 3 :L_1 :L_2)
:L_1
(x_1 <- i)
(x_1 >>= 1)
(x_1 *= 4)
(x_1 += a)
(aval <- (mem x_1 4))
(newaval <- aval)
(y_0 <- (mem ebp -4))
(newaval += y_0)
(newaval -= 1)
(x_2 <- x_1)
((mem x_2 4) <- newaval)
(nexti <- i)
(nexti += 2)
(ecx <- a)
(edx <- nexti)
(eax <- (mem ebp -4))
(goto :inc)
:L_2
(eax <- 1)
(return)
```

and this coloring


a
i
esi
len
edi
altlen
ebx
x_1
eax
aval
edx
newaval
ecx
y_0
x_2
nexti

(but it would have been better to use the callee save rewrite to spill one those variables)