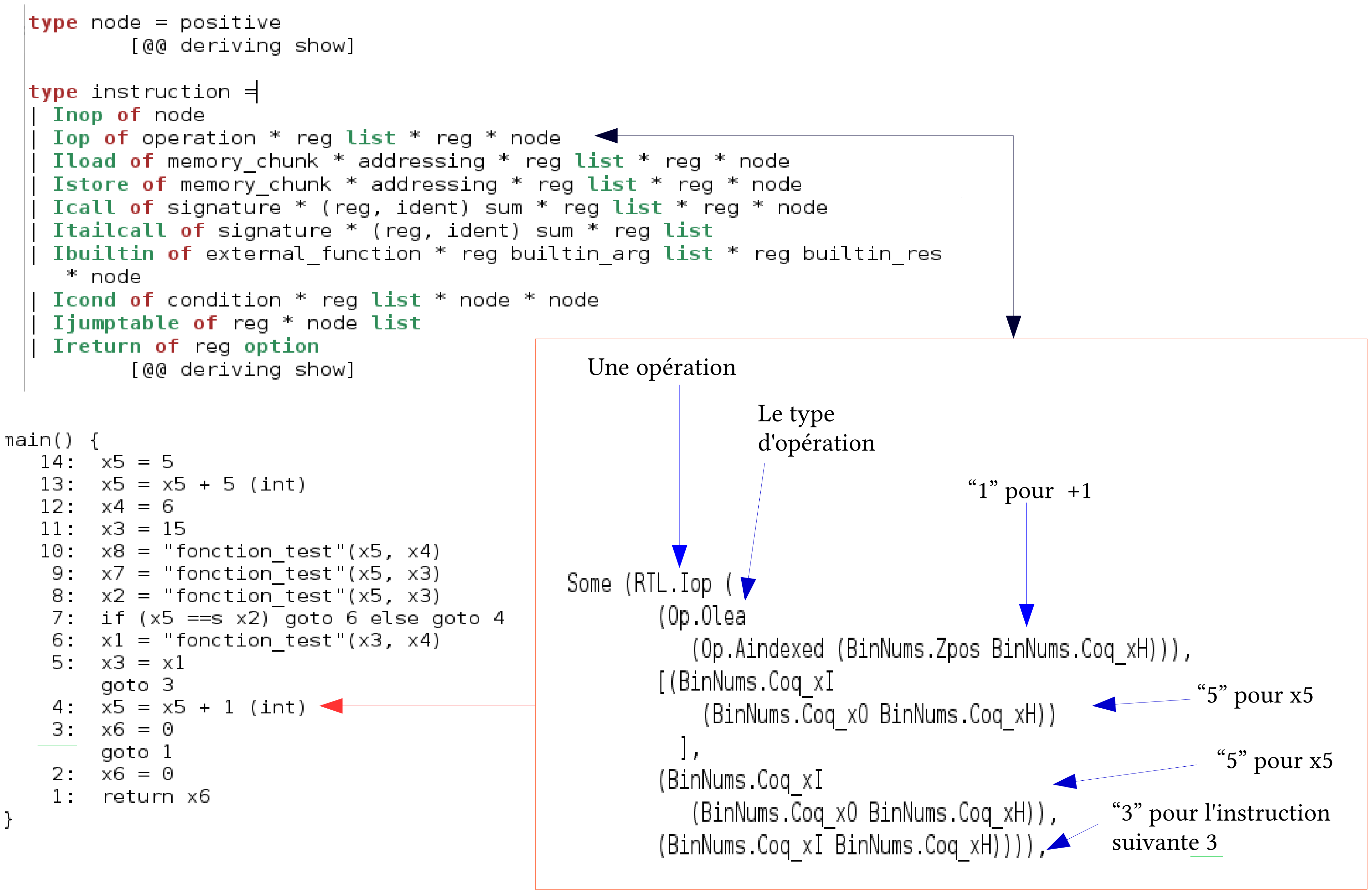

```
type node = positive
        [@@ deriving show]

type instruction =
| Inop of node
| Iop of operation * reg list * reg * node
| Iload of memory_chunk * addressing * reg list * reg * node
| Istore of memory_chunk * addressing * reg list * reg * node
| Icall of signature * (reg, ident) sum * reg list * reg * node
| Itailcall of signature * (reg, ident) sum * reg list
| Ibuiltin of external_function * reg builtin_arg list * reg builtin_res
  * node
| Icond of condition * reg list * node * node
| Ijumptable of reg * node list
| Ireturn of reg option
        [@@ deriving show]
```

```
main() {
   14:  x5 = 5
   13:  x5 = x5 + 5 (int)
   12:  x4 = 6
   11:  x3 = 15
   10:  x8 = "fonction_test"(x5, x4)
    9:  x7 = "fonction_test"(x5, x3)
    8:  x2 = "fonction_test"(x5, x3)
    7:  if (x5 ==s x2) goto 6 else goto 4
    6:  x1 = "fonction_test"(x3, x4)
    5:  x3 = x1
        goto 3
    4:  x5 = x5 + 1 (int)
    3:  x6 = 0
        goto 1
    2:  x6 = 0
    1:  return x6
}
```

```
Some (RTL.Iop (
       (Op.Olea
          (Op.Aindexed (BinNums.Zpos BinNums.Coq_xH))),
       [(BinNums.Coq_xI
           (BinNums.Coq_x0 BinNums.Coq_xH))
        ],
       (BinNums.Coq_xI
          (BinNums.Coq_x0 BinNums.Coq_xH)),
       (BinNums.Coq_xI BinNums.Coq_xH)))),
```