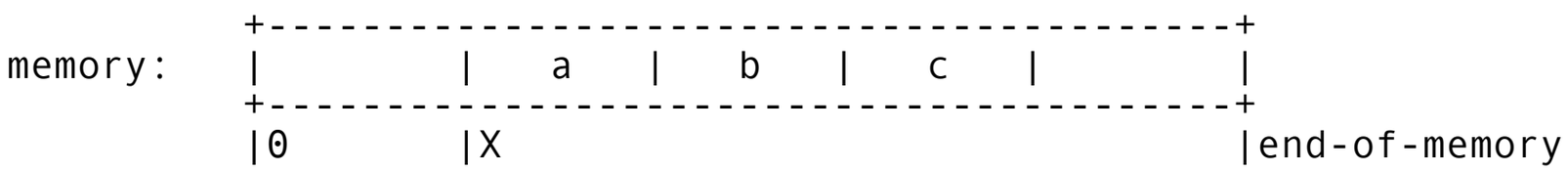
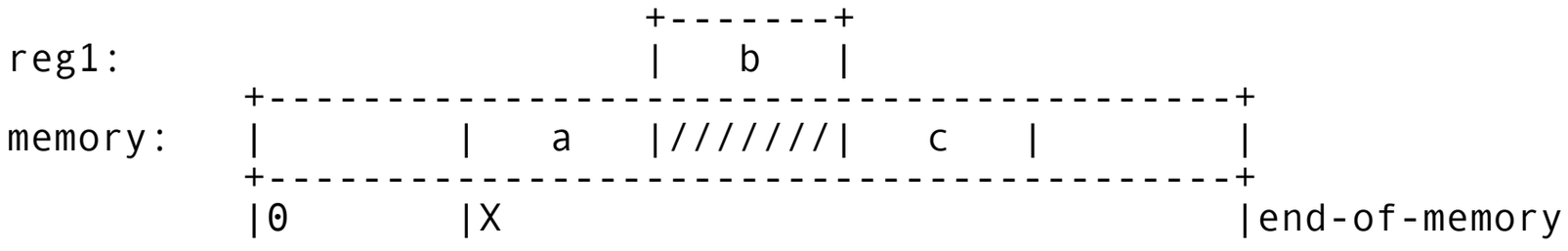
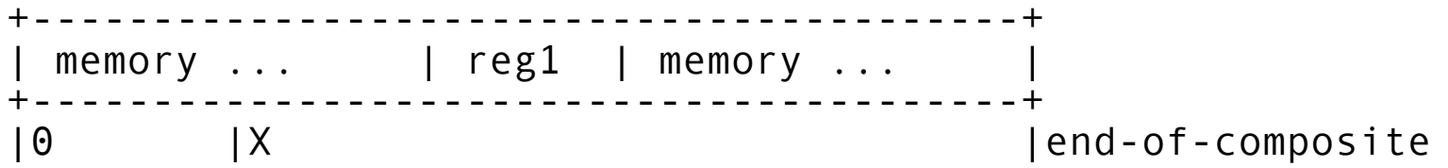
As a simple example, consider the location of a variable of a struct type with three 32-bit fields, whose default location is in memory at address X:



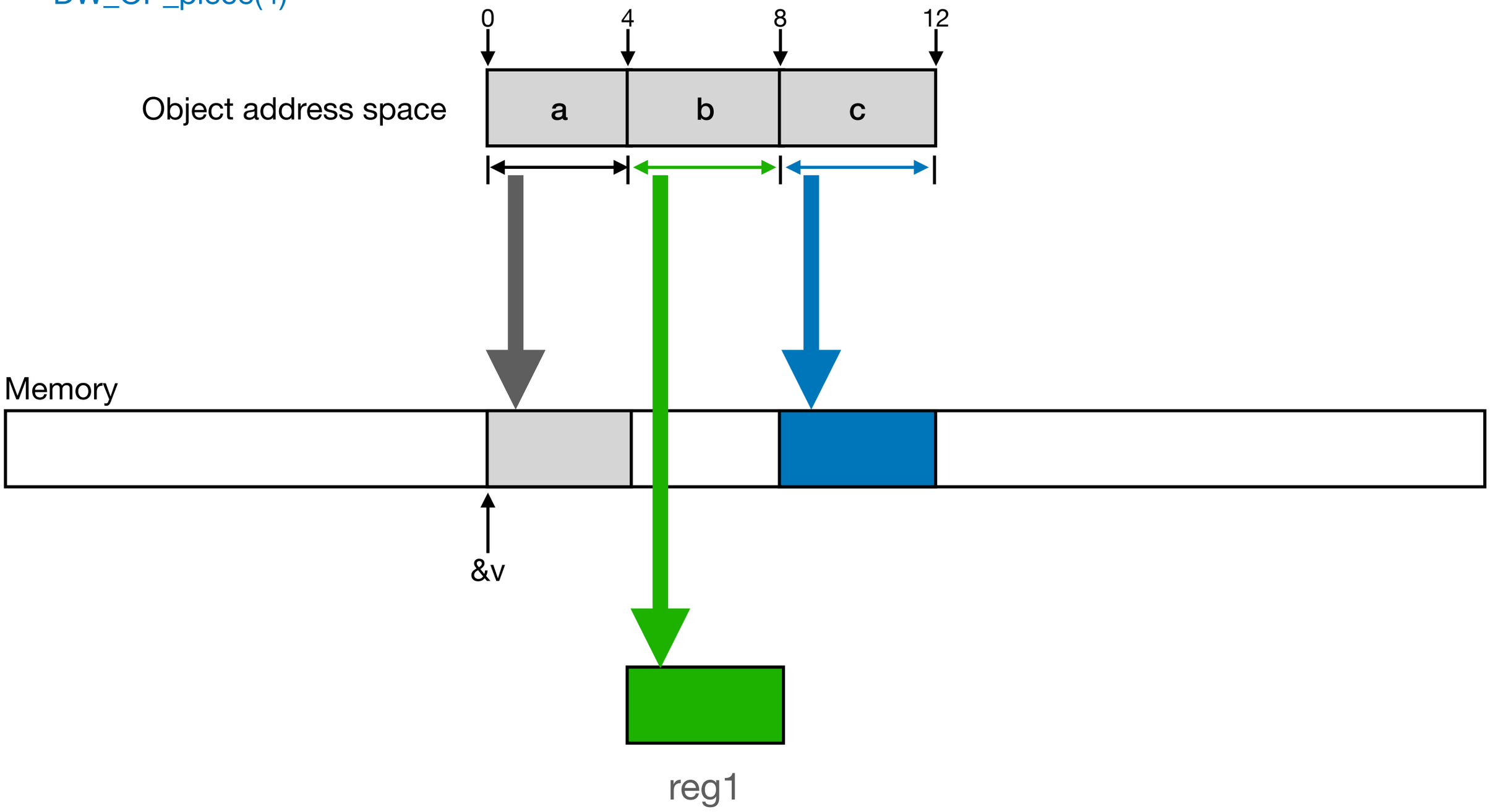

If the compiler chooses to promote field `b` to `reg1` for some section of code, we would describe this as an overlay on top of the memory location, of 4 bytes starting at offset 4 - relative to X - of the `reg1` location.



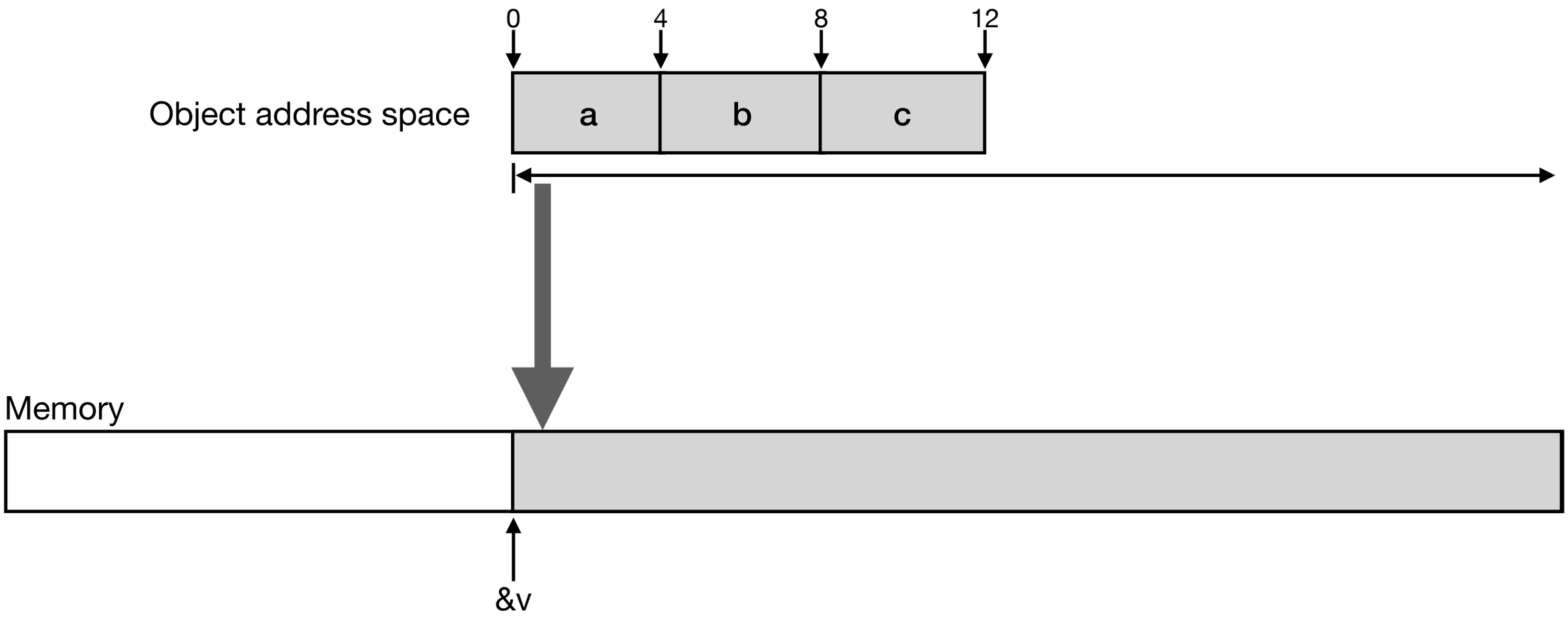

Resulting in the following composite location with three pieces:



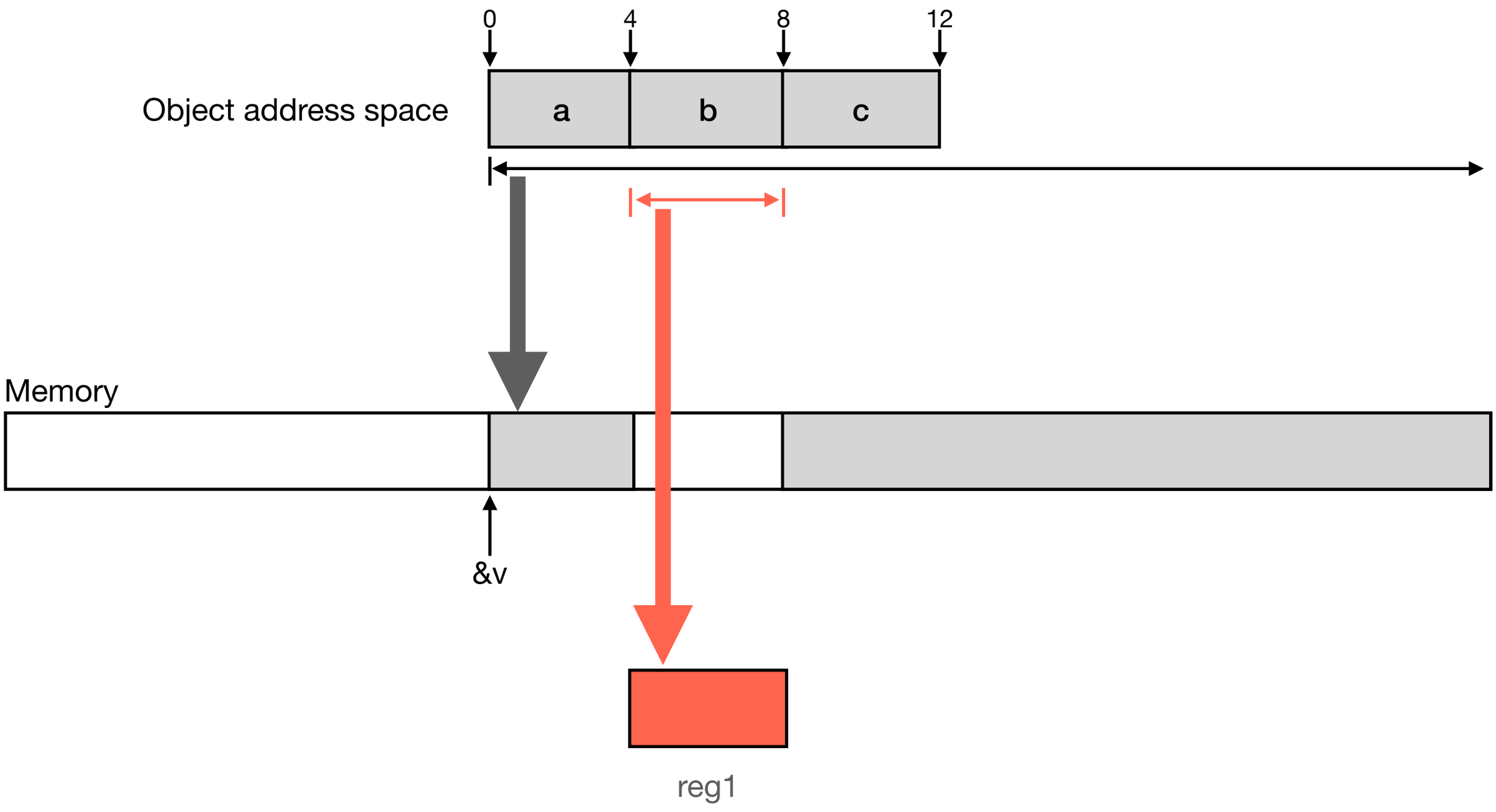
DW_OP_composite
DW_OP_addr(&v)
DW_OP_piece(4)
DW_OP_reg1
DW_OP_piece(4)
DW_OP_addr(&v+8)
DW_OP_piece(4)



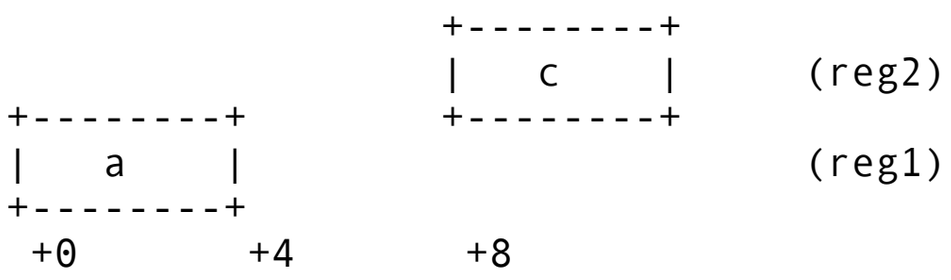
DW_OP_addr(&v)
DW_OP_reg1
DW_OP_lit4
DW_OP_lit4
DW_OP_overlay



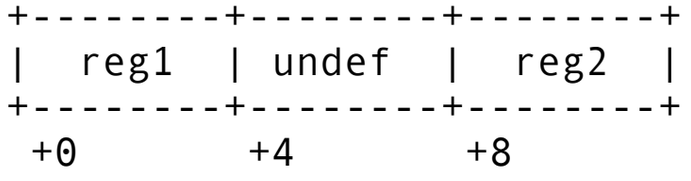
DW_OP_addr(&v)
DW_OP_reg1
DW_OP_lit4
DW_OP_lit4
DW_OP_overlay



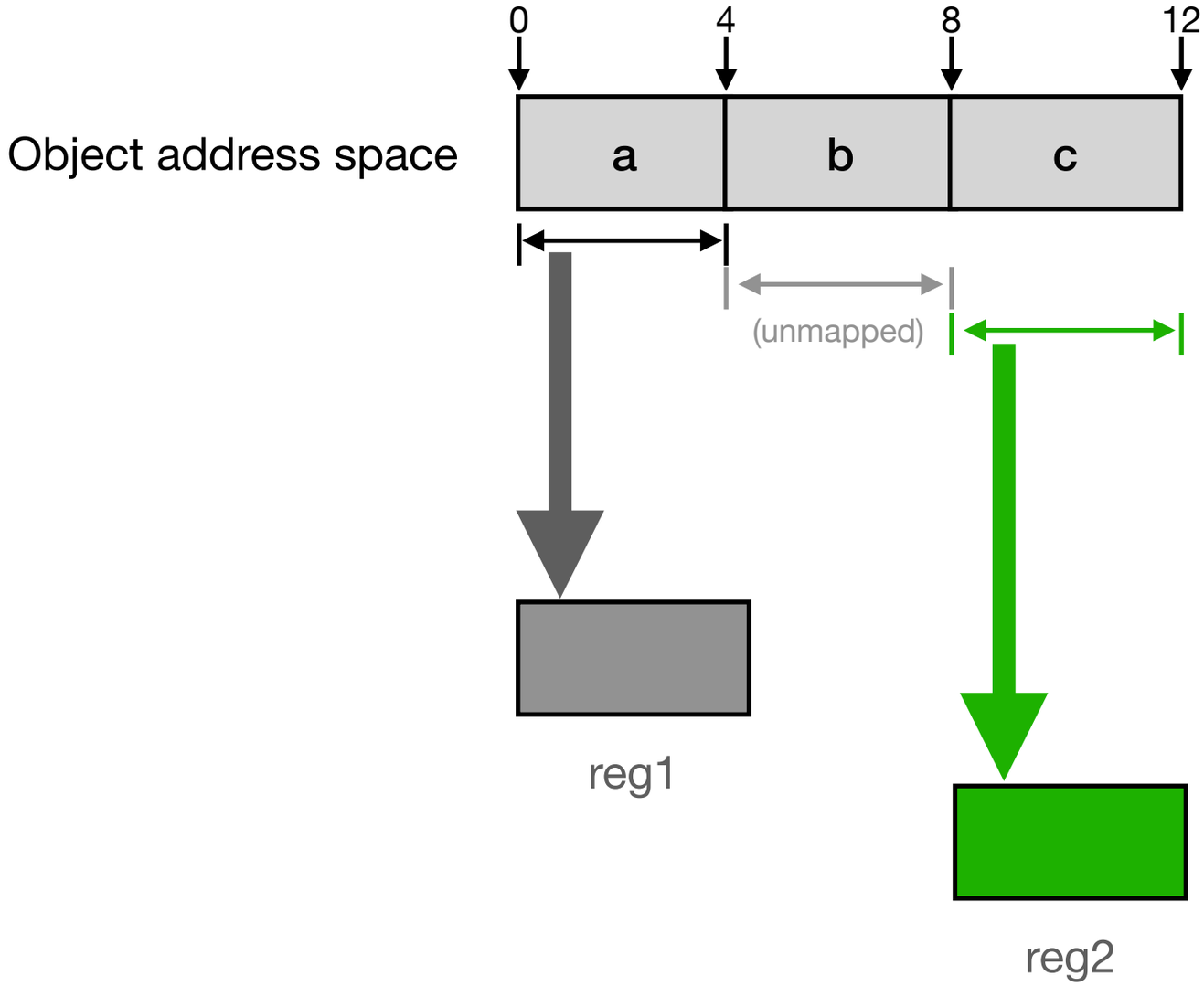
If the compiler chooses to promote fields a and c to registers, and optimize out field b, we may describe this with concatenation with a gap, like so:



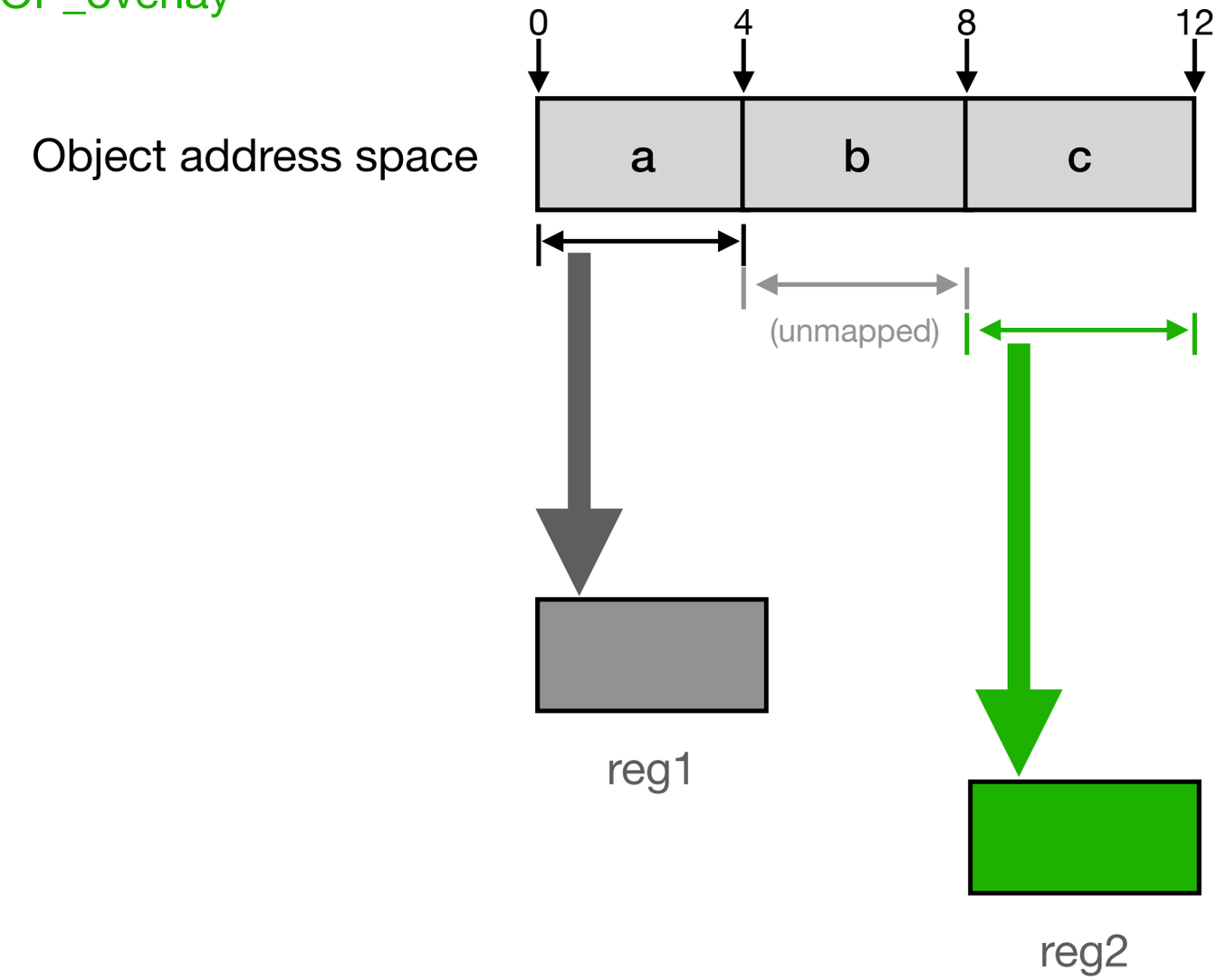

This results in the following composite:



DW_OP_reg1
DW_OP_reg2
DW_OP_lit8
DW_OP_lit4
DW_OP_overlay



DW_OP_composite  // or DW_OP_undefined
DW_OP_reg1
DW_OP_lit0
DW_OP_lit4
DW_OP_overlay
DW_OP_reg2
DW_OP_lit8
DW_OP_lit4
DW_OP_overlay



```
DW_OP_breg0(0)    // Address of array dst is in reg0
DW_OP_regx(vreg2)    // Vector register vreg2 holds elements dst[i]…
DW_OP_breg1(0)    // Loop index i is in reg1
DW_OP_lit4
DW_OP_mul    //compute offset of dst[i]
DW_OP_const1u(32)    // The size of the vector register
DW_OP_overlay    // Make a composite location description
```

