a[i] — *a

a[0] — *(a+i)

&a[i] — a+i

Match them!

int32_t* a

&expr

expr must be an L-value

L-Value: anything that can go on the left-hand side of =

x = v;

s.x = v;

a[i] = v;

s[i].x = v;

*a = v;

[(1+2) = v ; f() = v] not lvalues

High-level idea: each malloc stores metadata about allocated block

header
- even number representing a free block
- odd number representing a malloc'd, allocated, busy block

in both cases, the number with LSB set to 0 is the block size in bytes

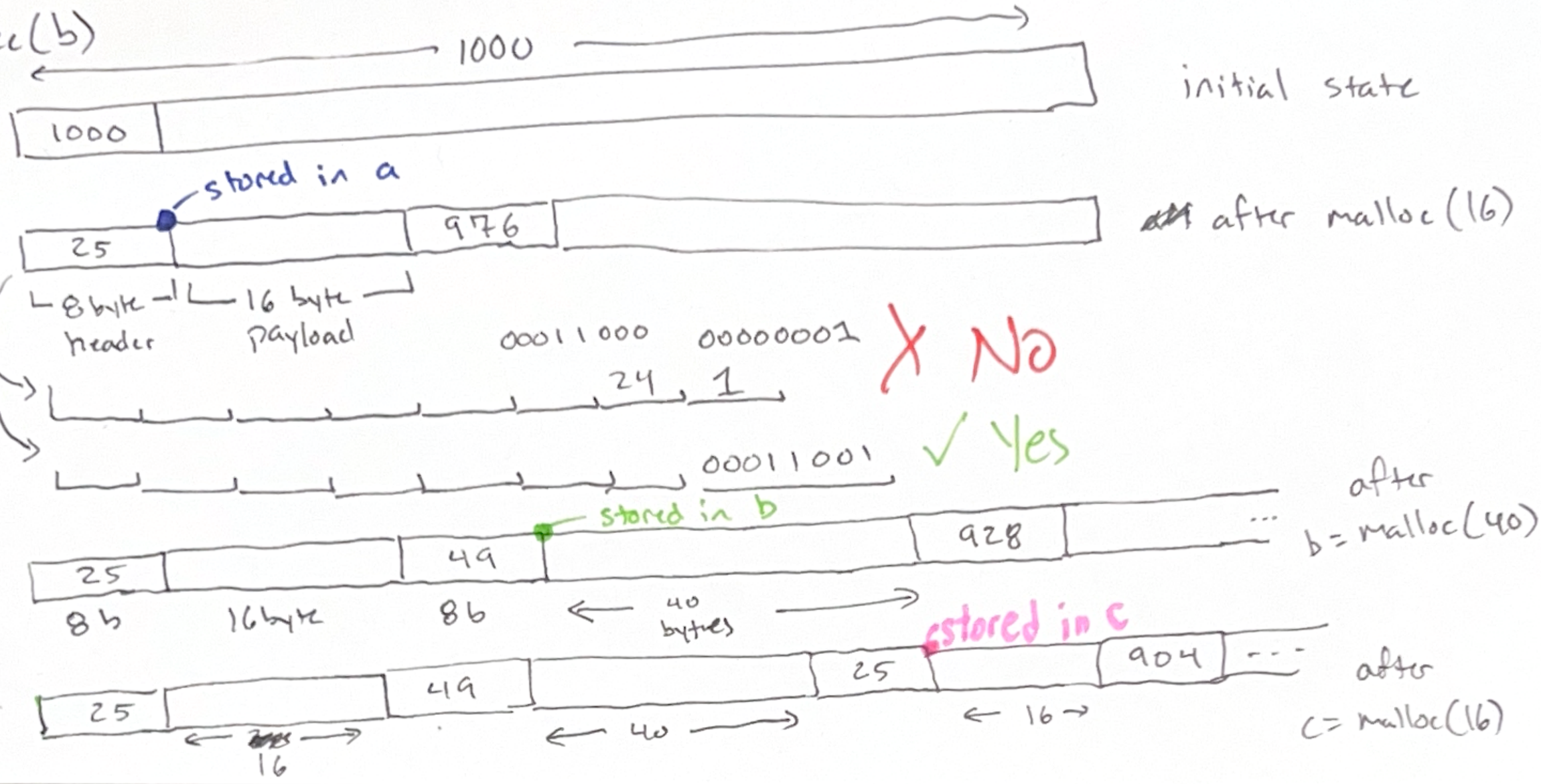
header = 33 allocated block of 32 bytes (24 byte payload)

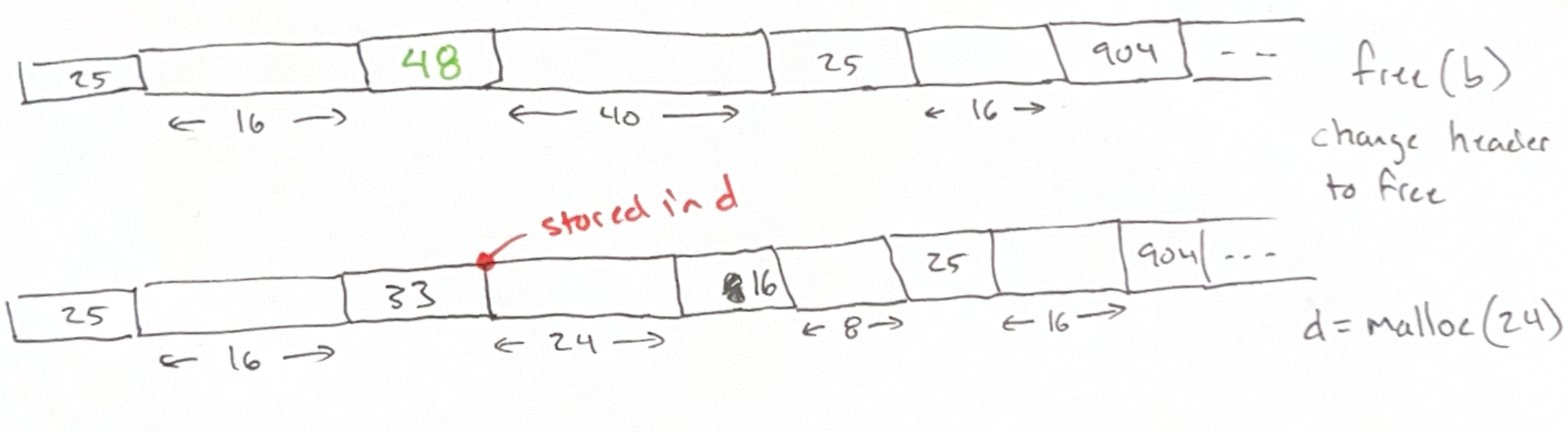
header = 108 free block of 108 bytes (100 byte payload)

"size of block" = full size including header

always make block sizes divisible by 8 (or 8-byte aligned)

a = malloc(16)
b = malloc(40)
c = malloc(16)
free(b)





"looping through the heap" – start at heap start and
increment a pointer in block-sized chunks