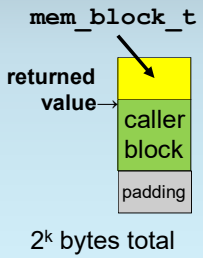
Actual Allocation Contains Three Sections

In other words, the block that we actually allocate includes
- a **`mem_block_t`**,
- **bytes for the caller**, and
- **padding** to a power of 2.

The pointer that we **return** is the **address of the caller's data** (after **`mem_block_t`**).



Linked List Heads are a File-Scope Array

The linked lists are lists of **`mem_block_t`**.

What do our bins look like in **C**?

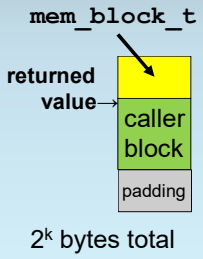
```
#define MEM220_MAX_ALLOC_LOG 20

static mem_block_t*
    mem_bin[MEM220_MAX_ALLOC_LOG+1];
```

The **head of the linked list**
- with **blocks of 2^k bytes**
- **has array index k**.

C Allows Programmers to Hide Details from Compiler

Notice that,
- from the point of view
- of code that manages linked list of free blocks,
- only the **`mem_block_t`** exists.

The **blocks are generally larger than `sizeof (mem_block_t)`**.



Interface Offers Four Functions Similar to the C Library

Next, let's take a look at the API.

We'll **write four functions**
- corresponding to the four
- that we discussed
- in the **C** library.

```
void* mem220_allocate
        (size_t n_bytes);
```

You've seen this routine already.

It **behaves the same as `malloc`**.

