Malloc Lab

15/18-213: Introduction to Computer Systems
1011st Recitation, Apr. 2, 2012

Outline

- Overview
- Naïve Implicit List Allocator
- Getting Started
- Evaluation
- Debugging Tips, Misc

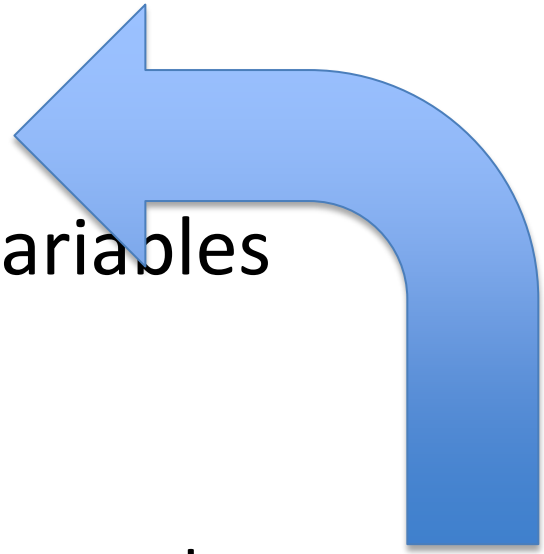
You will implement

- malloc()
- calloc()
- realloc()
- free()
- mm_init()
  - setup whatever you need to
  - called before any calls to malloc
- mm_checkheap()
  - a debug function

In other words, you will manage the heap yourself.

You have

- mem_sbrk()
- unused heap space
- a handful of *scalar* variables
  - no arrays
  - no structs
  - where will you keep your data structures?



mem_sbrk()

- a wrapper around sbrk()
  - sbrk(incr) asks the operating system for incr more bytes of heap space at the end of the current heap
  - has relatively high overhead

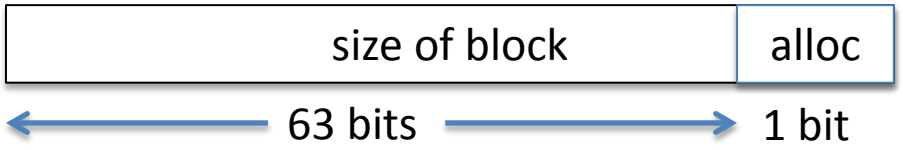
Outline

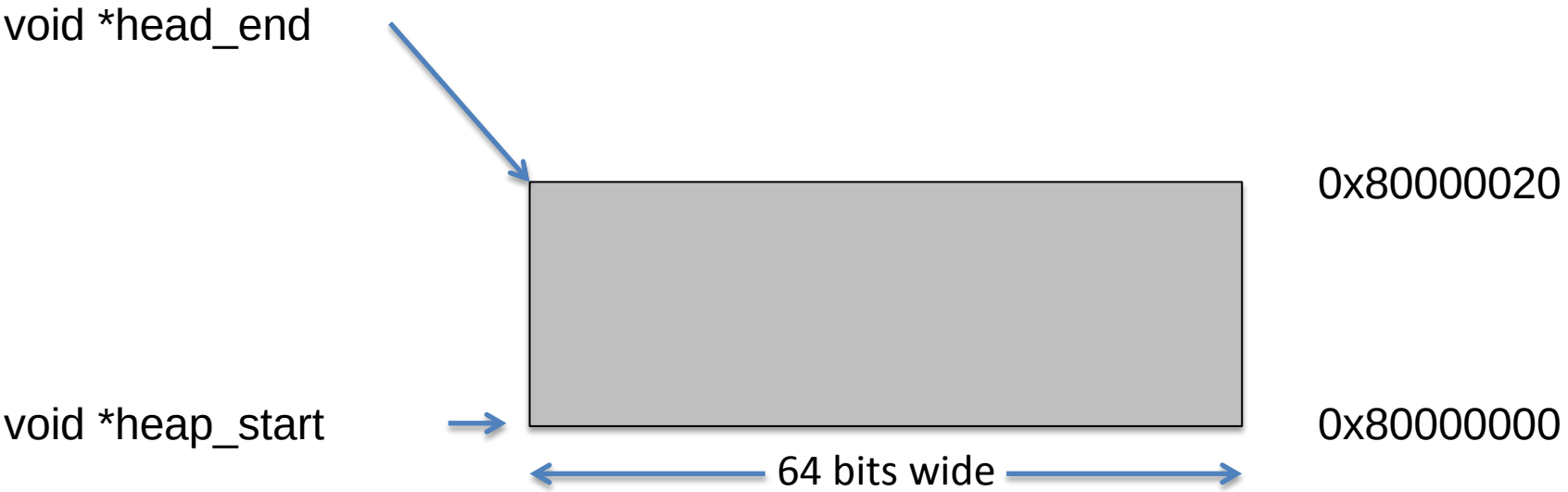
- ~~Overview~~
- Naïve Implicit List Allocator
- Getting Started
- Evaluation
- Debugging Tips, Misc

Before mm_init()

During mm_init()

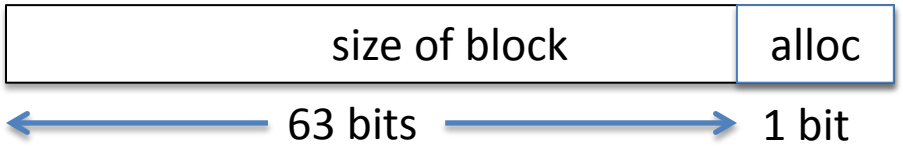
header/footer field format:

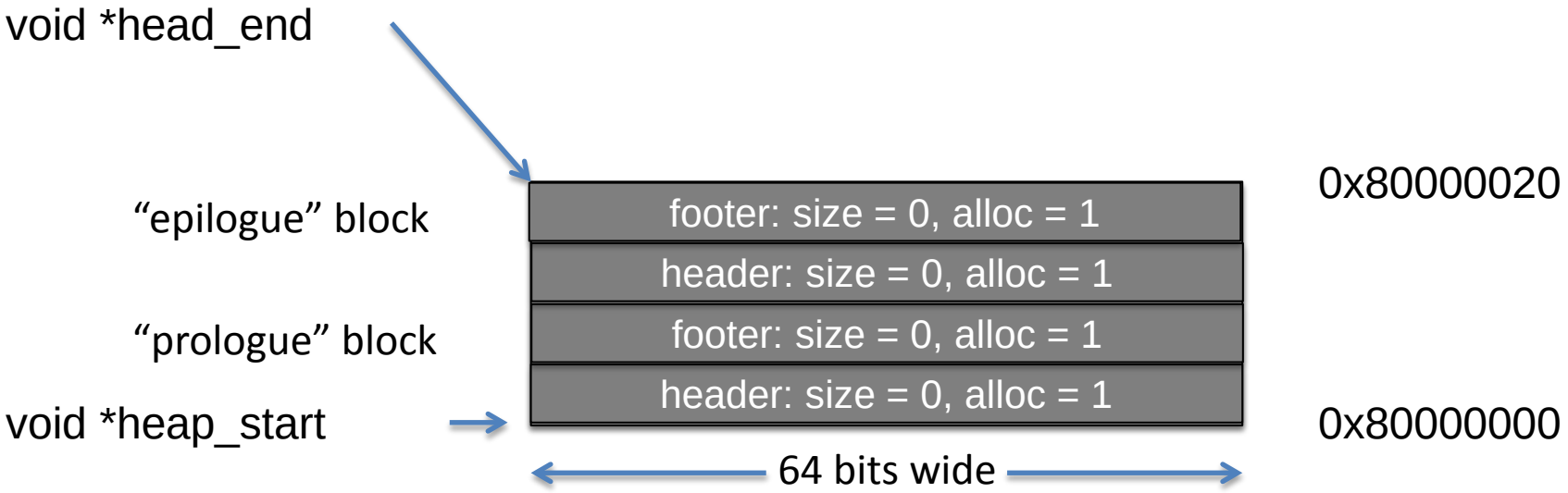




During mm_init()

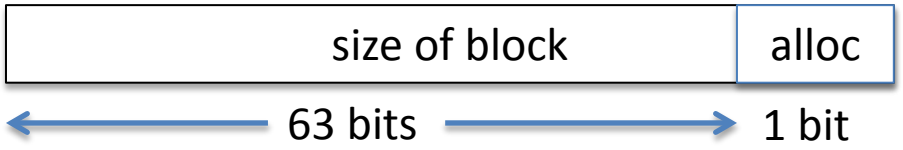
header/footer field format:



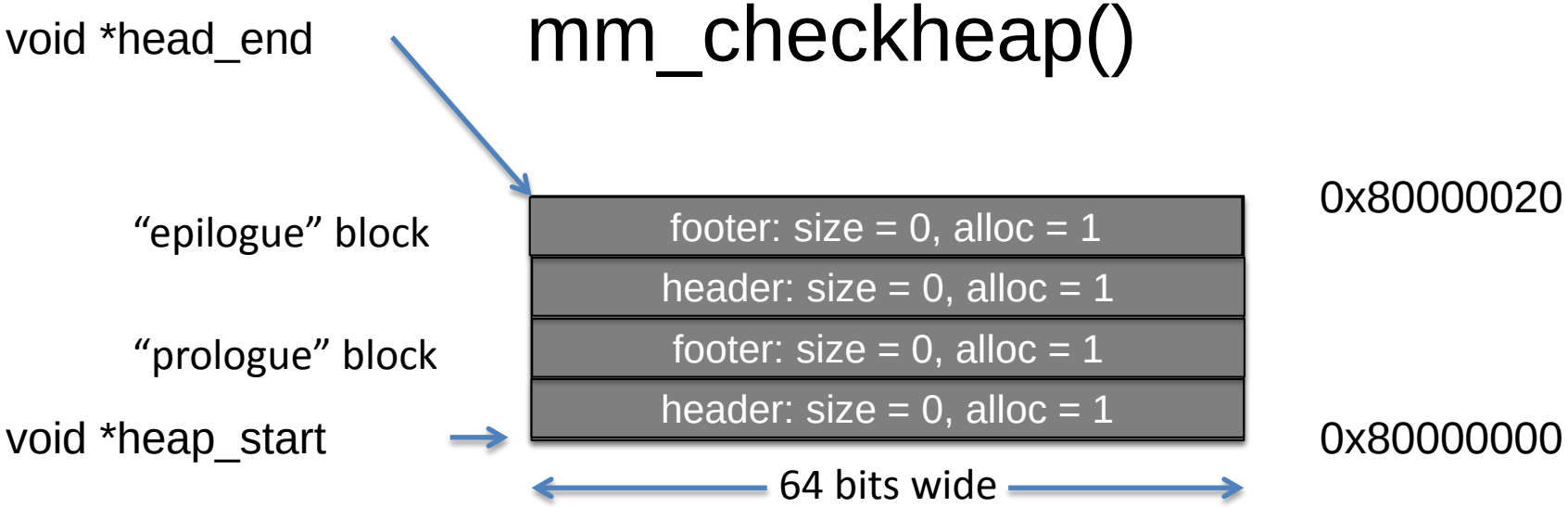


After mm_init()

header/footer field format:

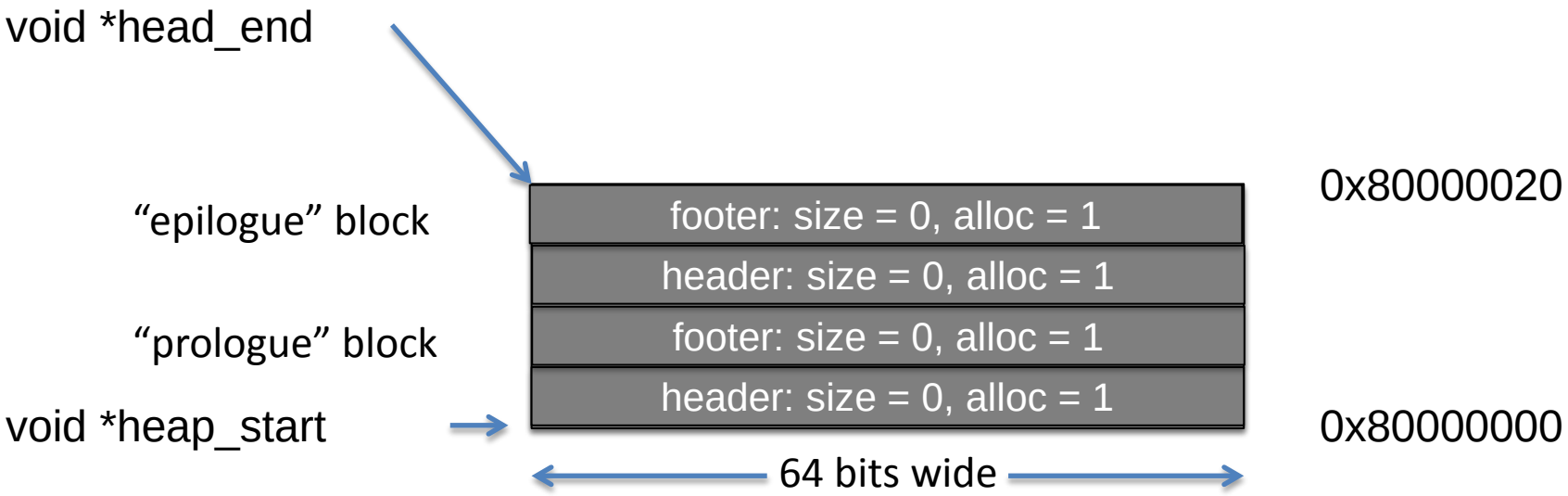


We've set up our invariants – at this point we can implement mm_checkheap()



Aside: mm_checkheap()

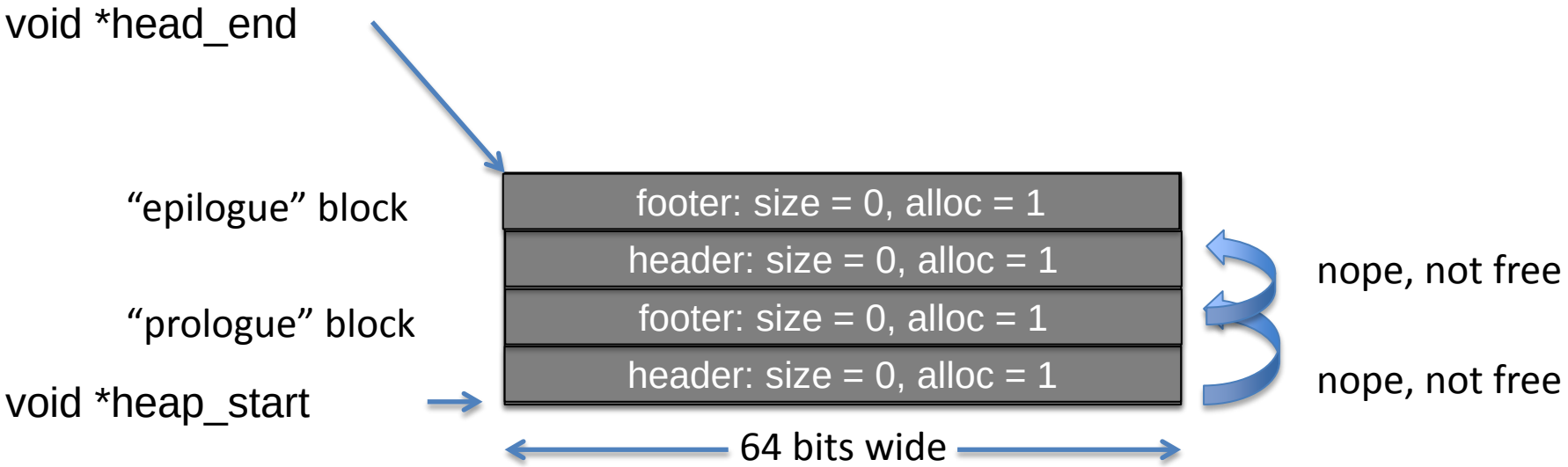
- mm_checkheap()
  - called between malloc()/free() operations
  - should check invariants are obeyed
  - quickly find bugs



Call to malloc(0x100)

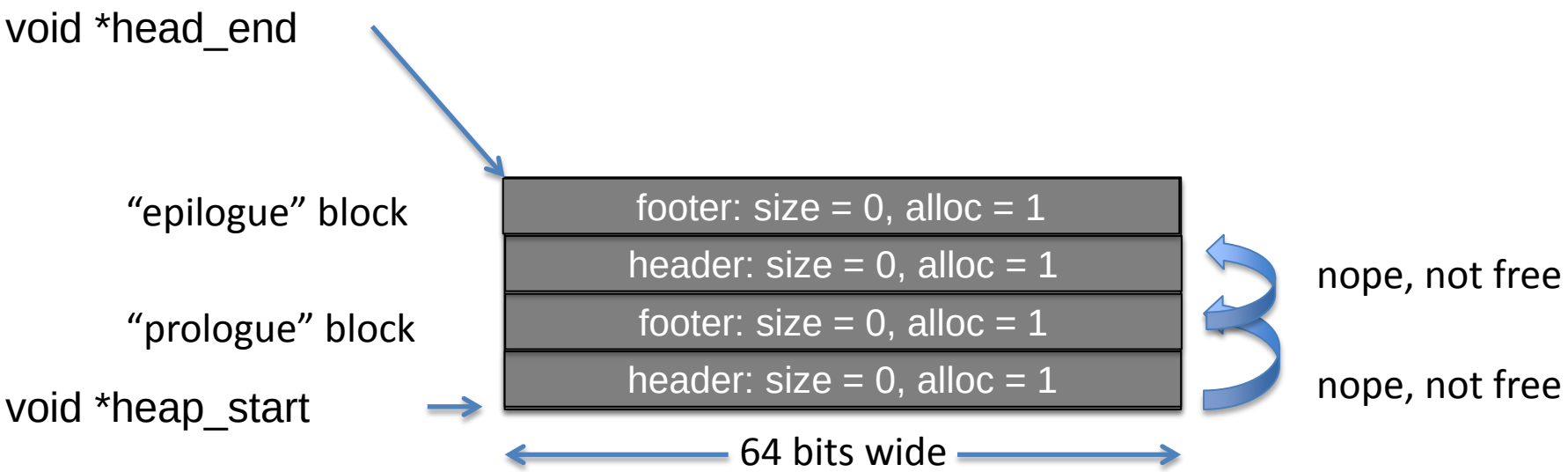
Search through heap for a free block

Use the fact that header = start of previous block's footer + 8 bytes

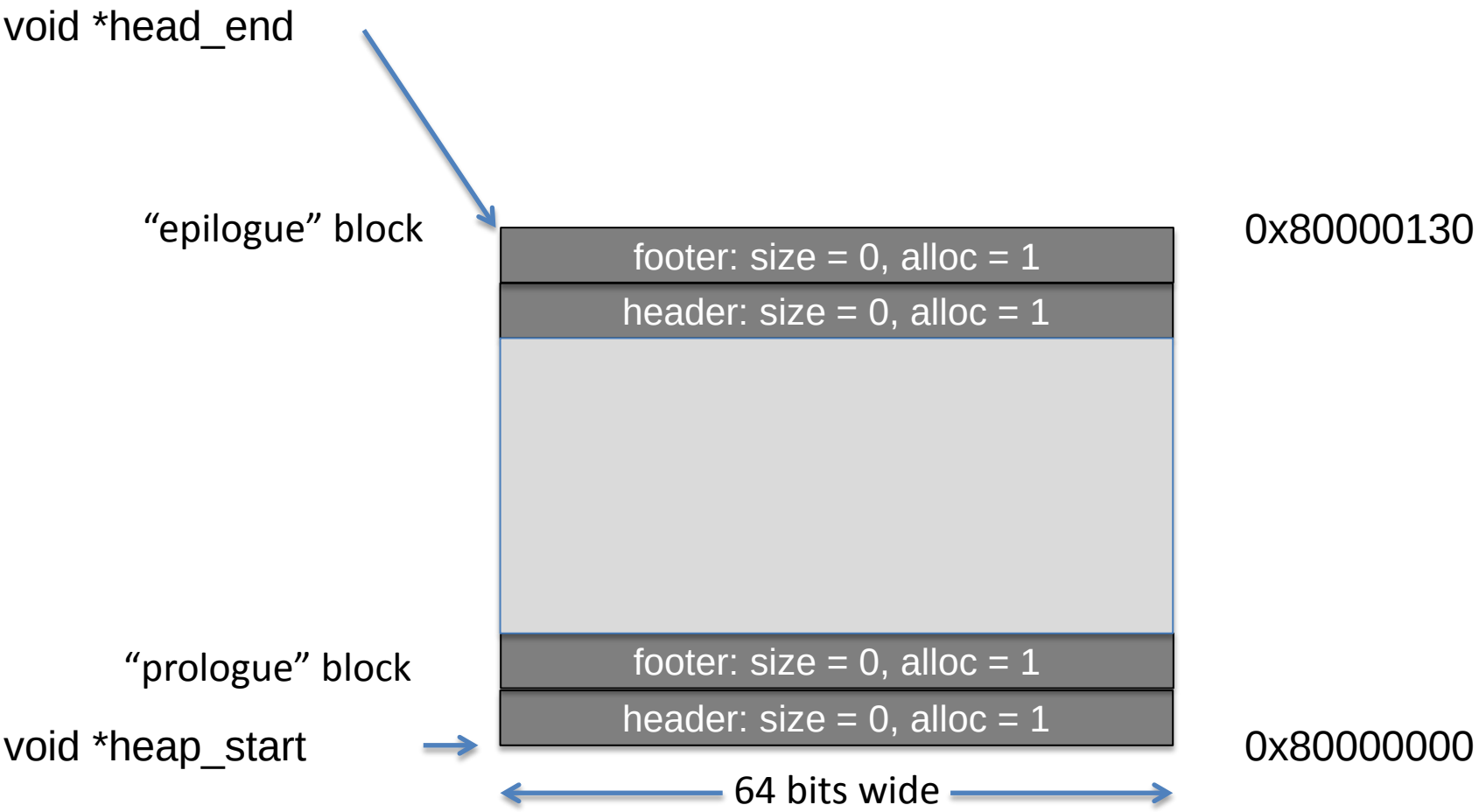


Call to malloc(0x100)

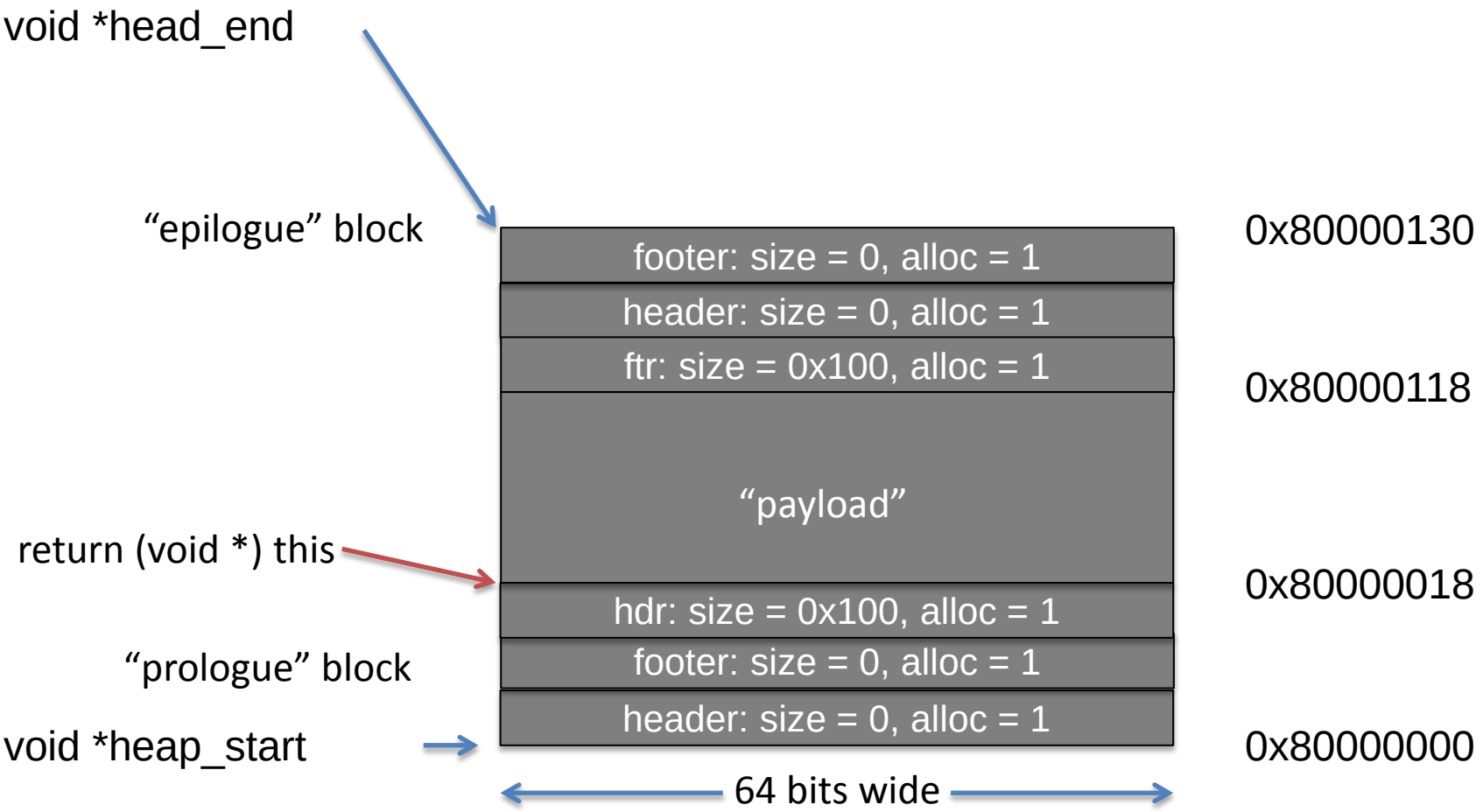
Search failed to find free block. Now what?



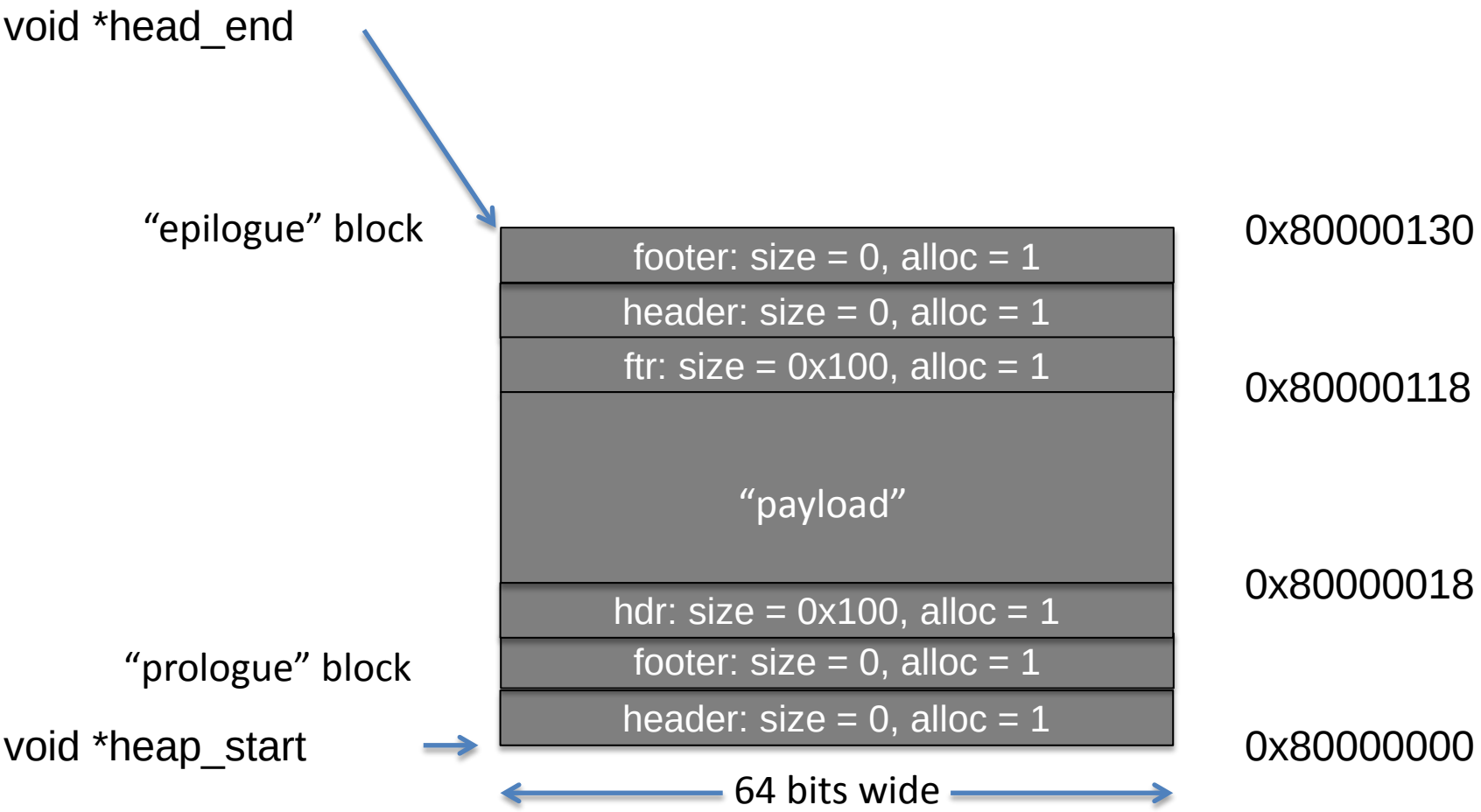
Call to malloc(0x100)



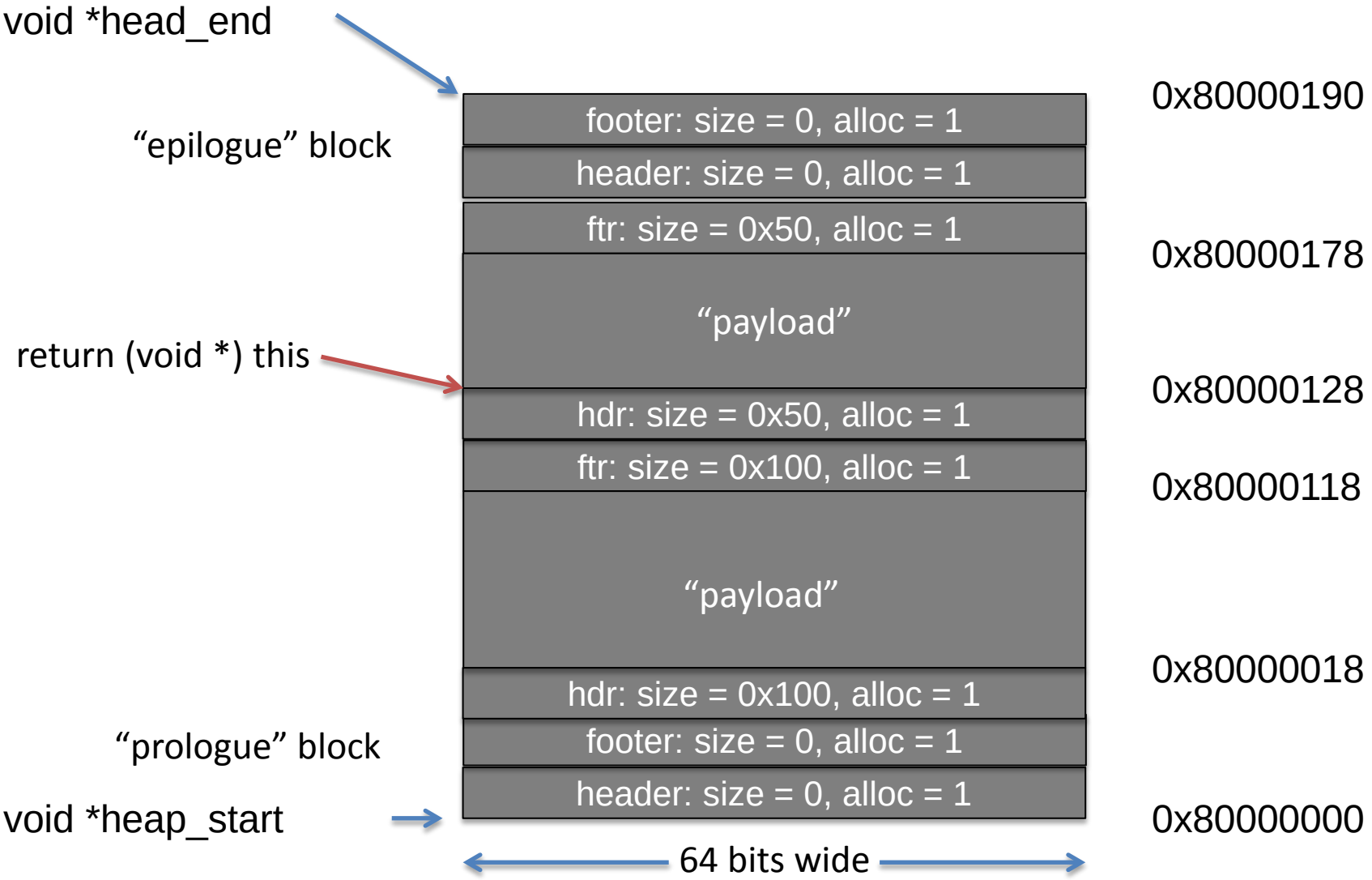
Call to malloc(0x100)



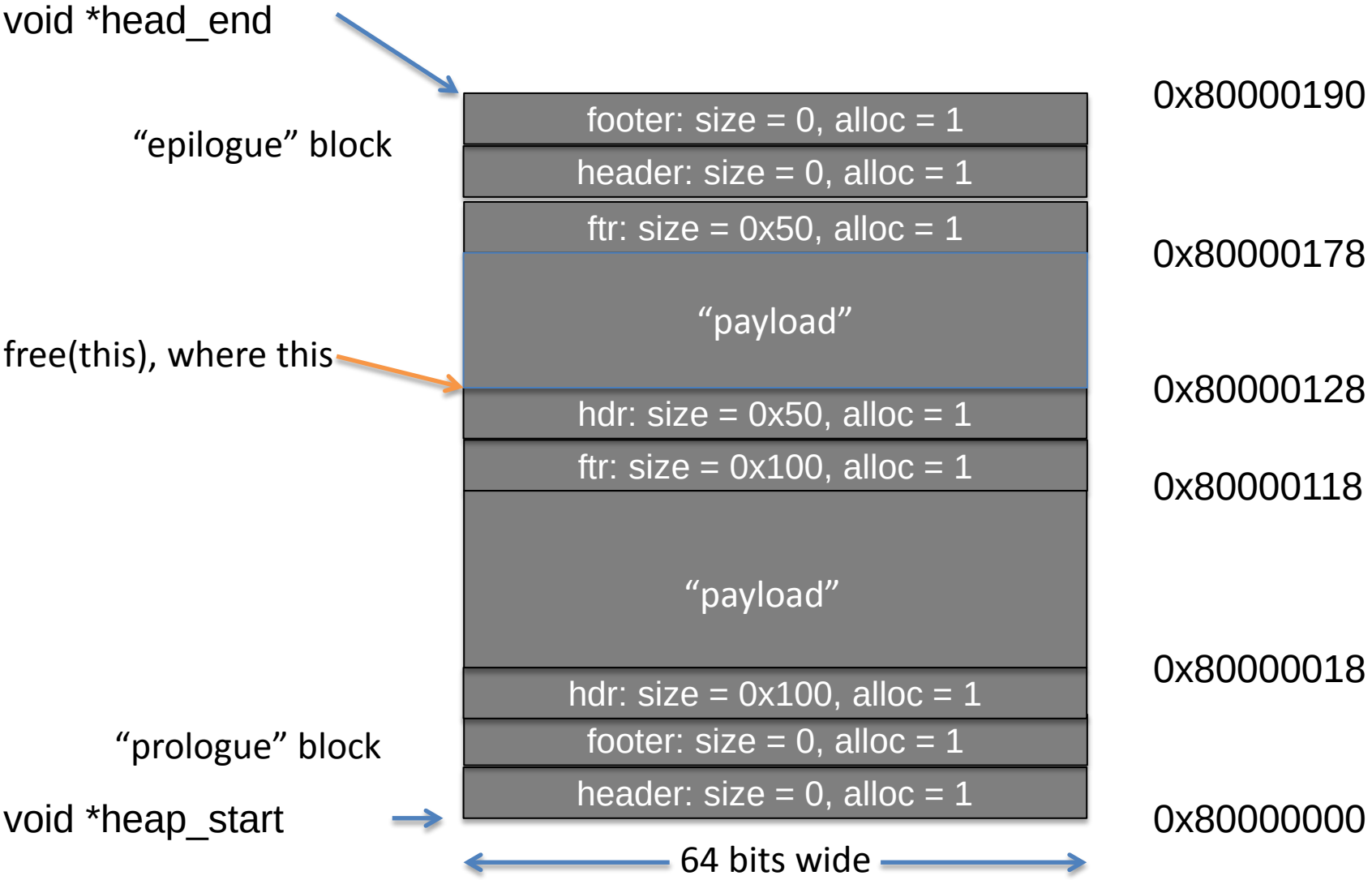
Now, Call to malloc(0x50)



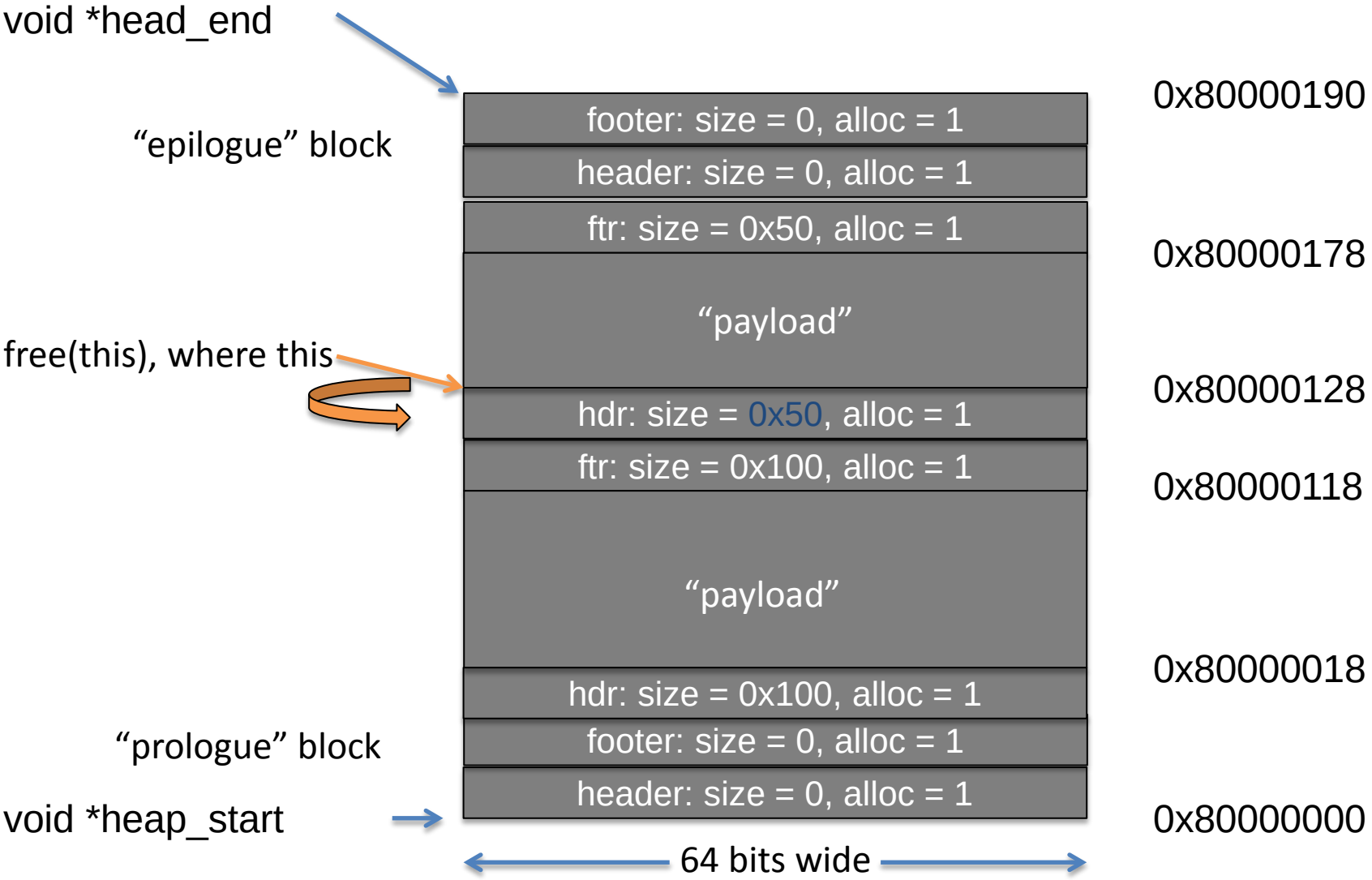
Now, Call to malloc(0x50)



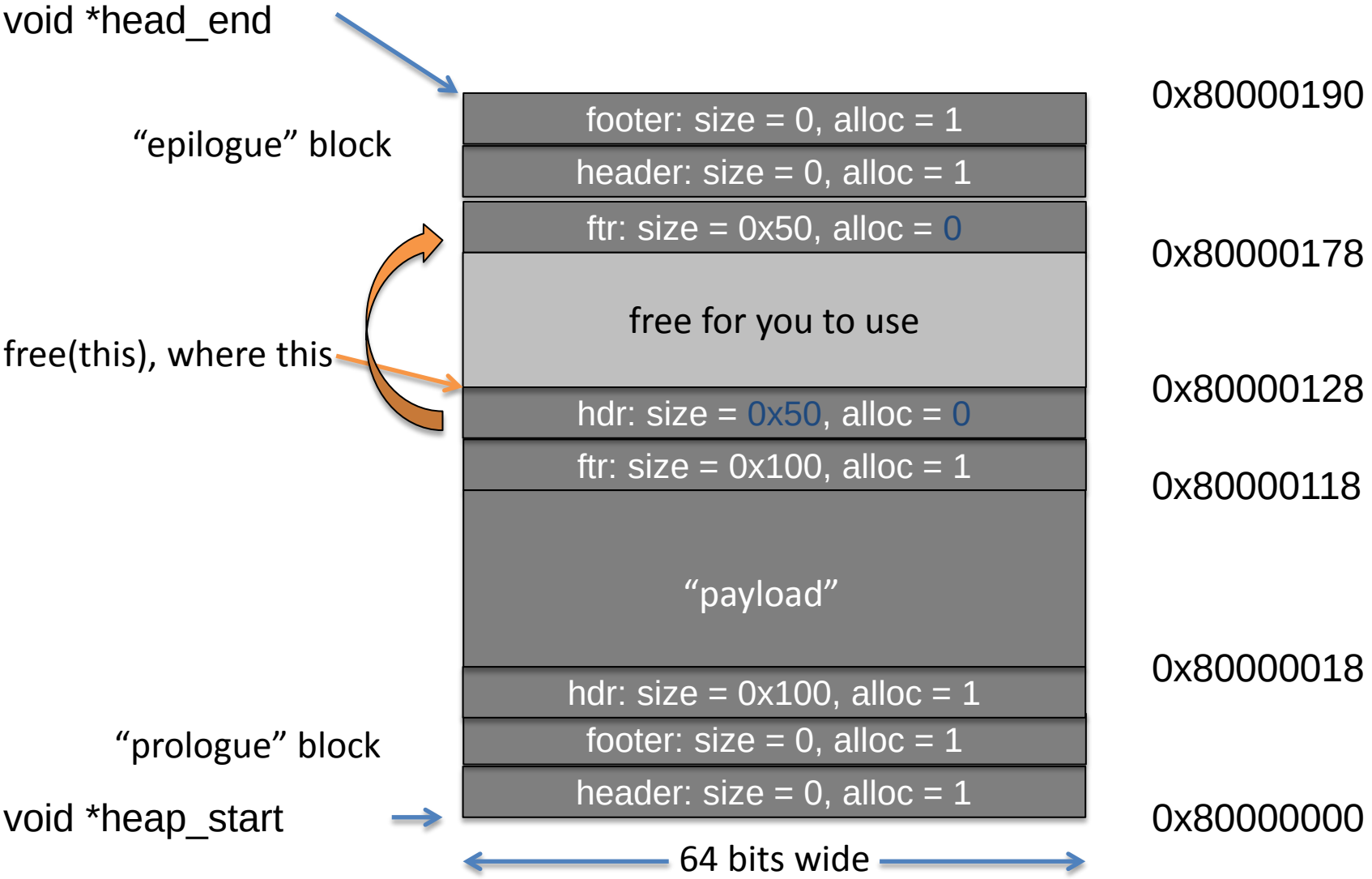
suddenly,free(0x80000128)



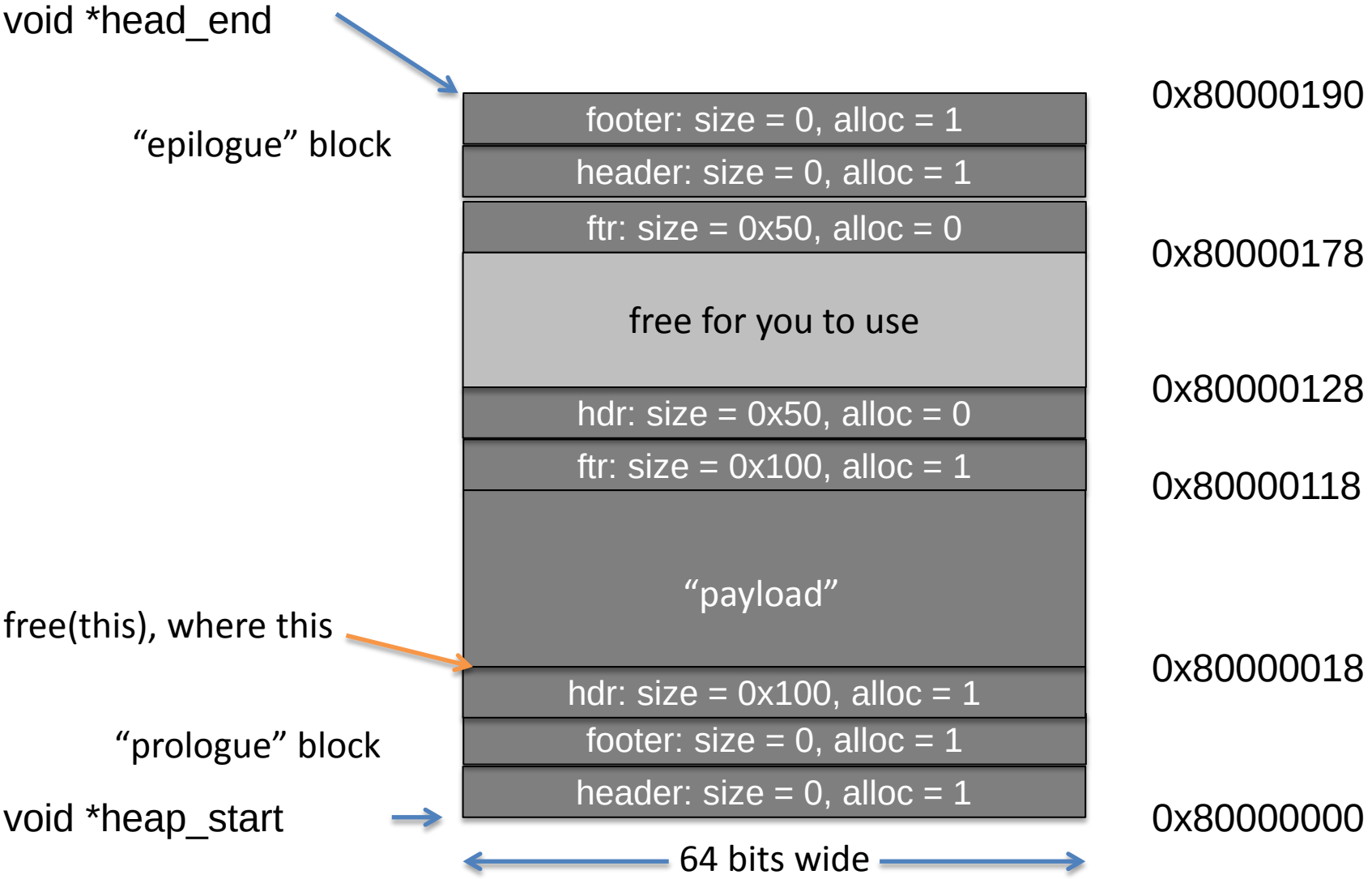
suddenly,free(0x80000128)



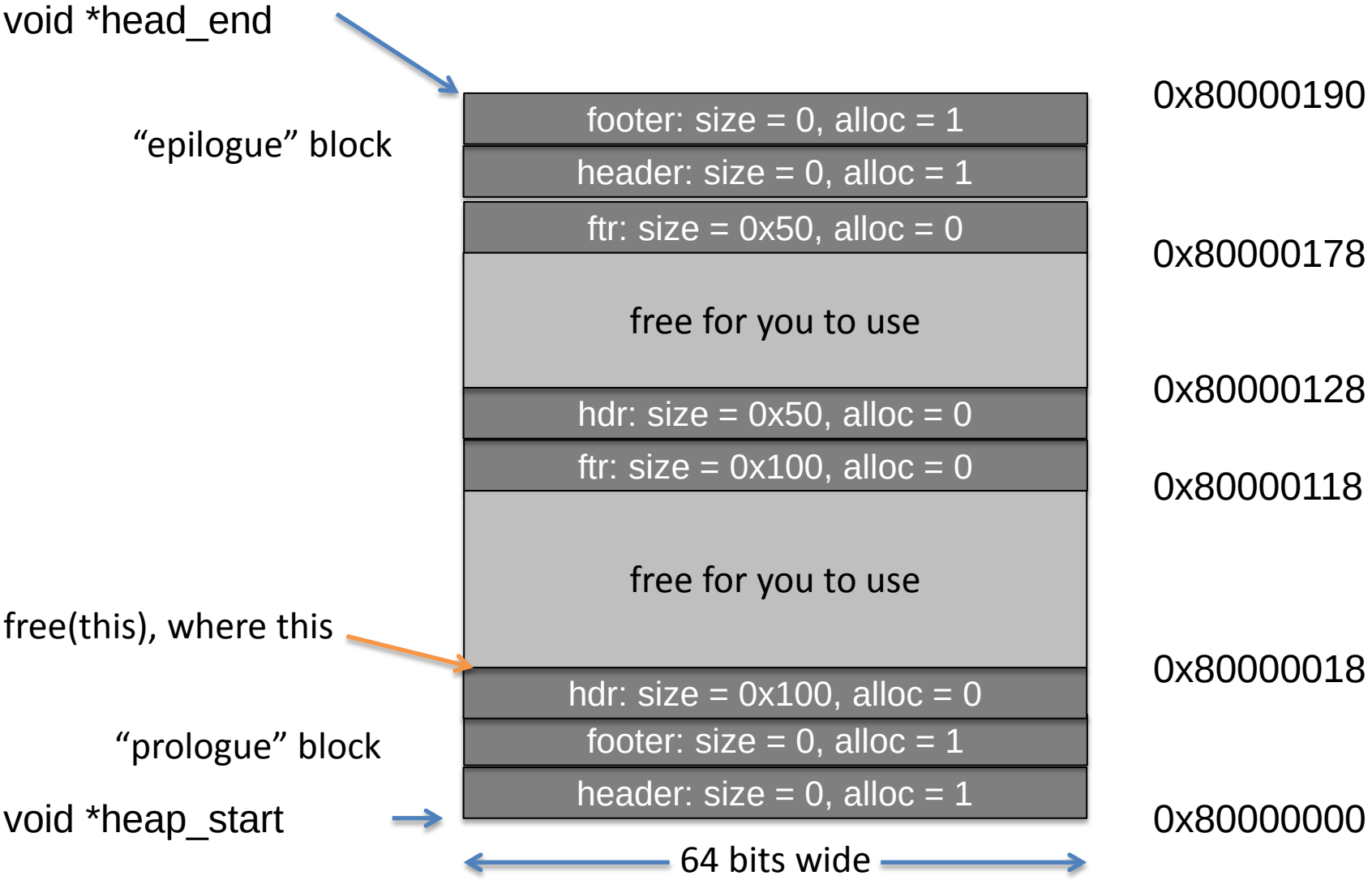
suddenly,free(0x80000128)



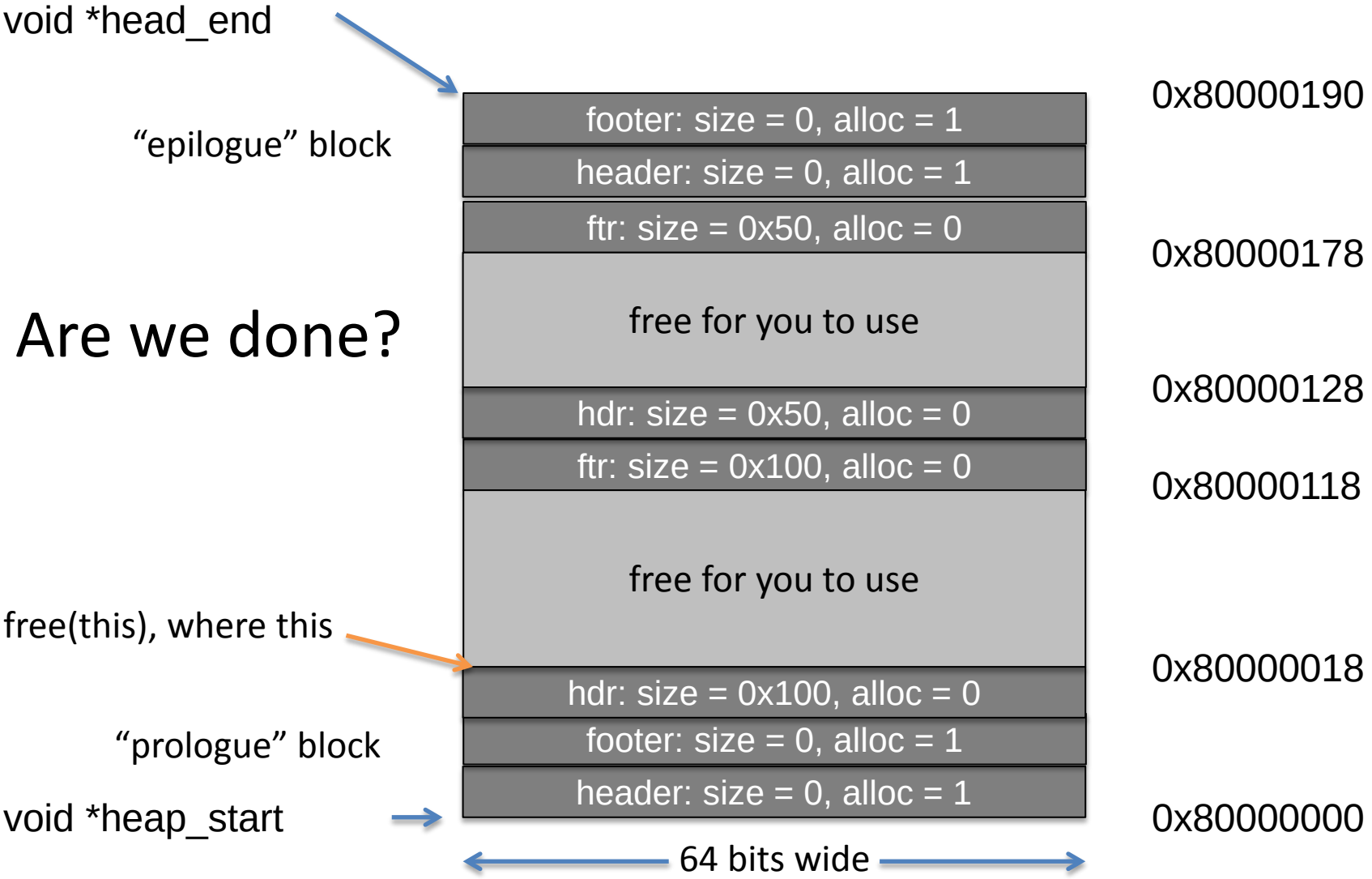
suddenly,free(0x80000018)



suddenly,free(0x80000018)

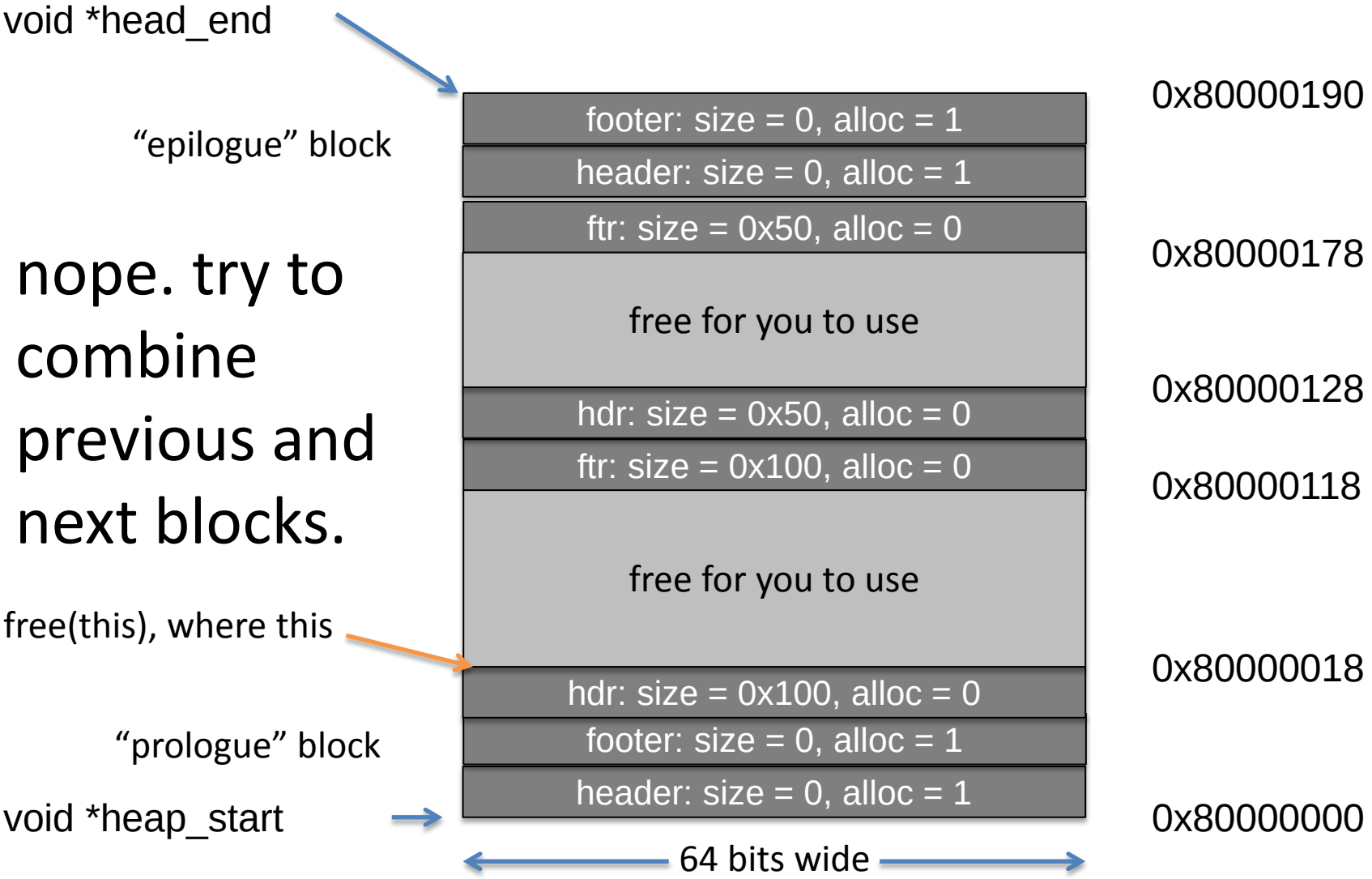


suddenly,free(0x80000018)

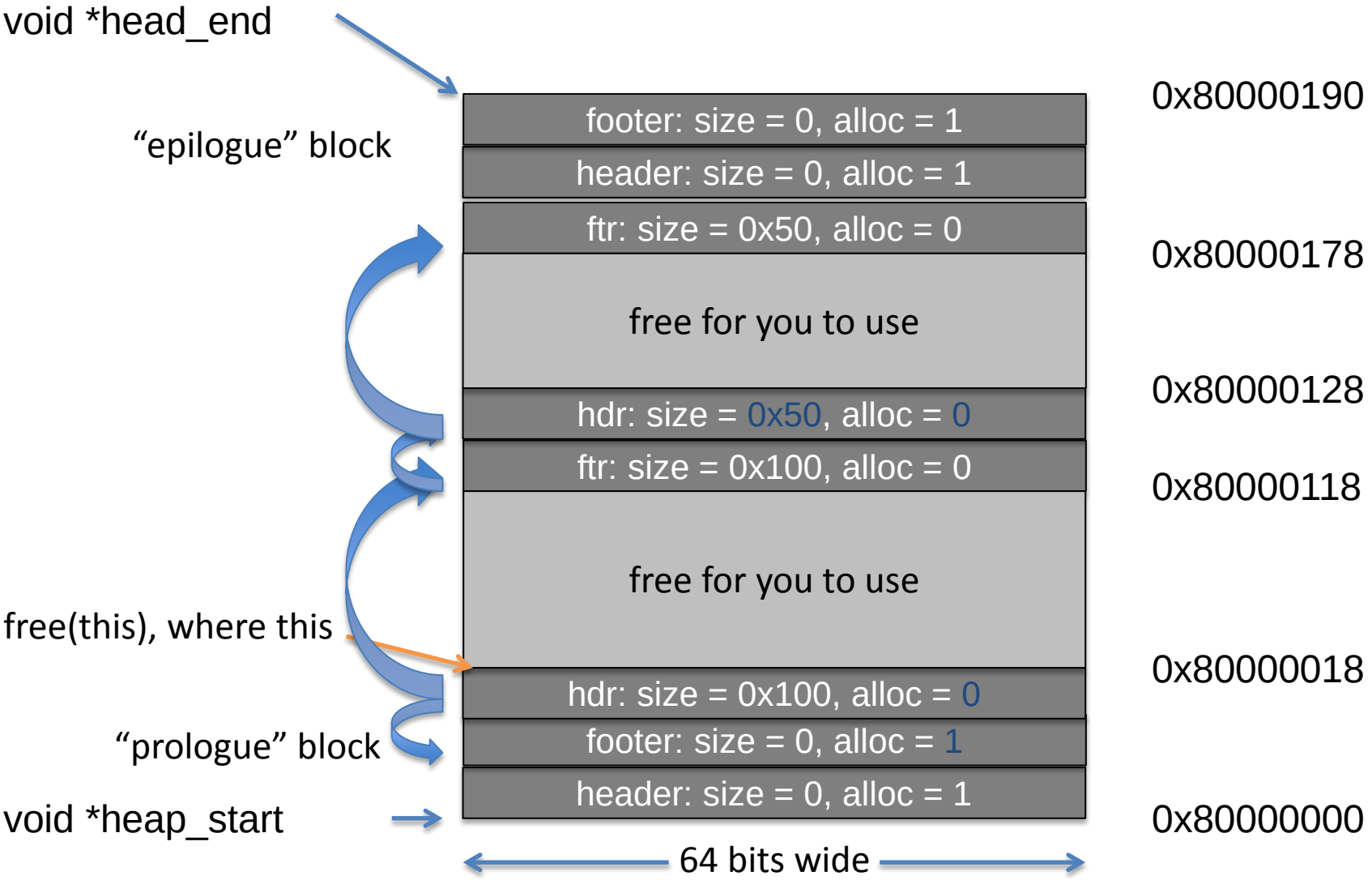


Are we done?

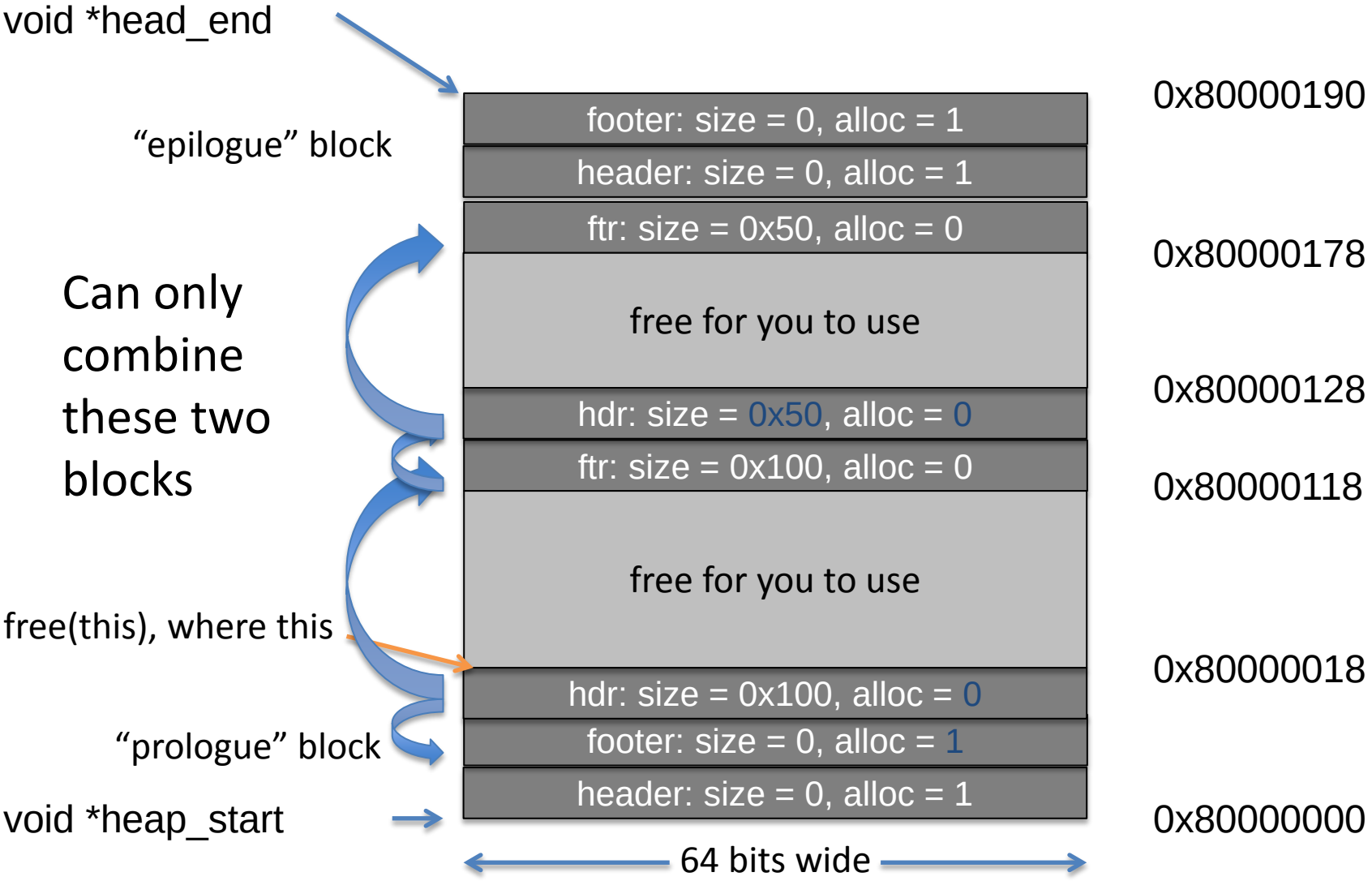
suddenly,free(0x80000018)



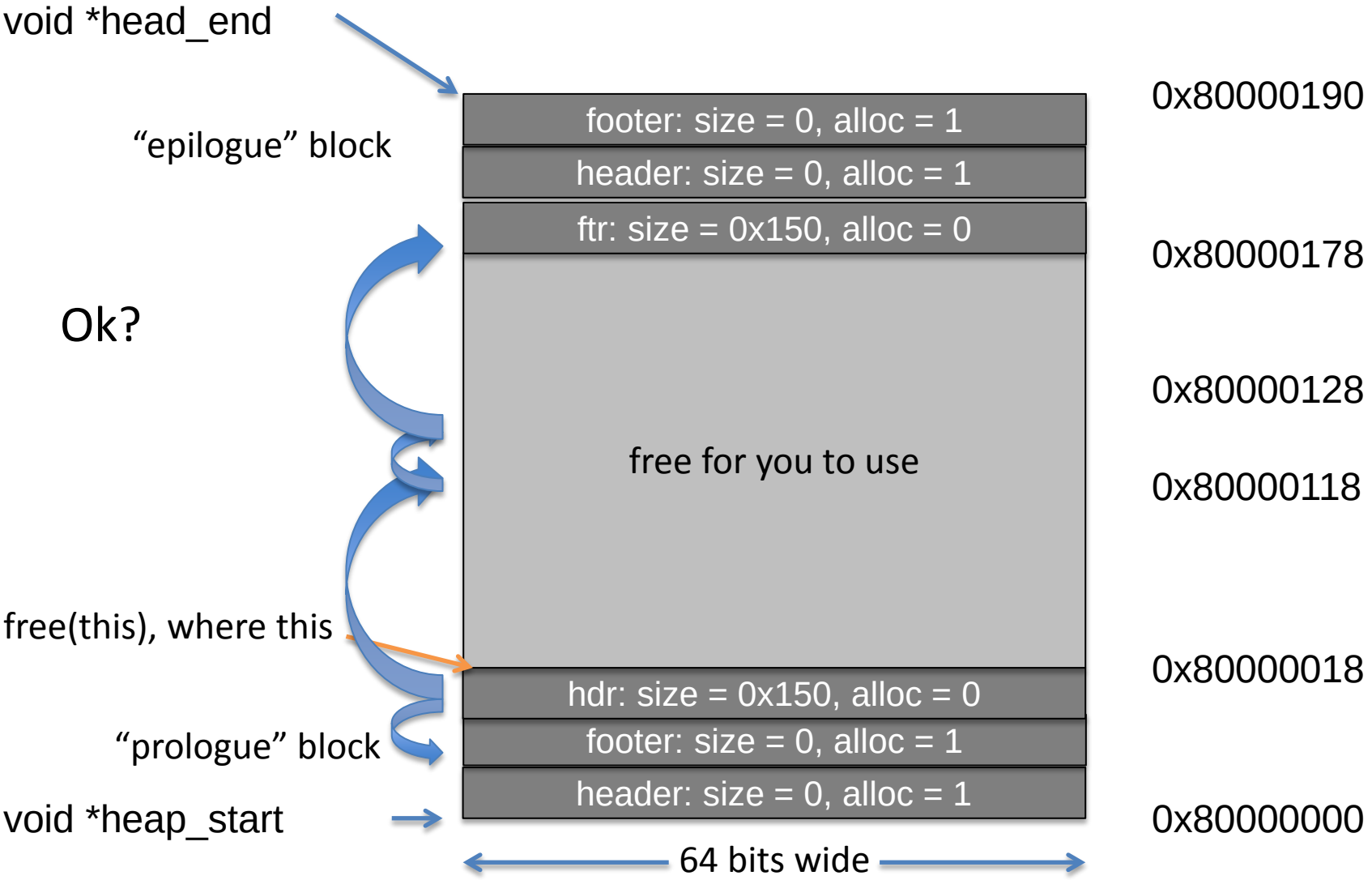
suddenly,free(0x80000018)



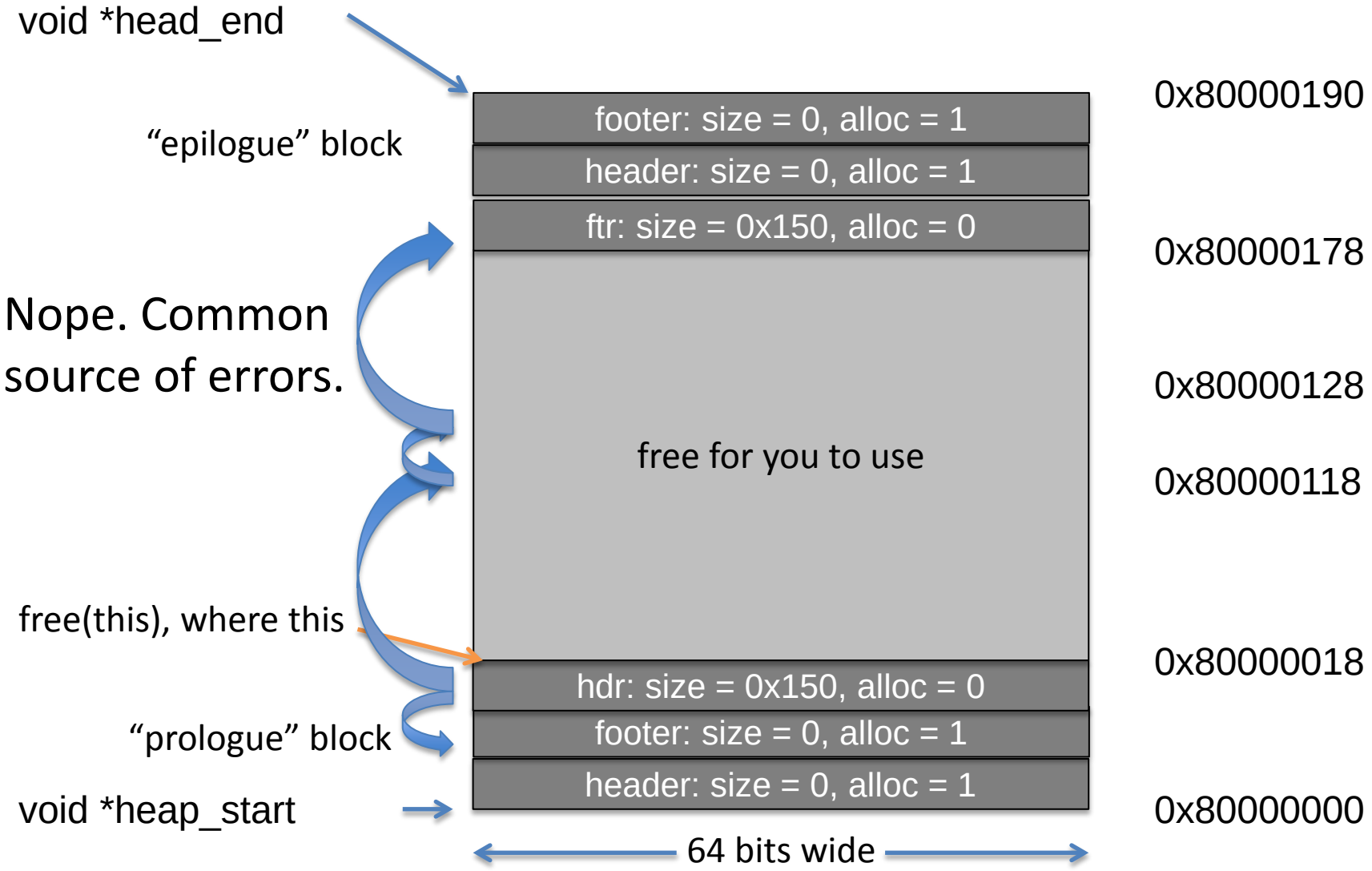
suddenly,free(0x80000018)



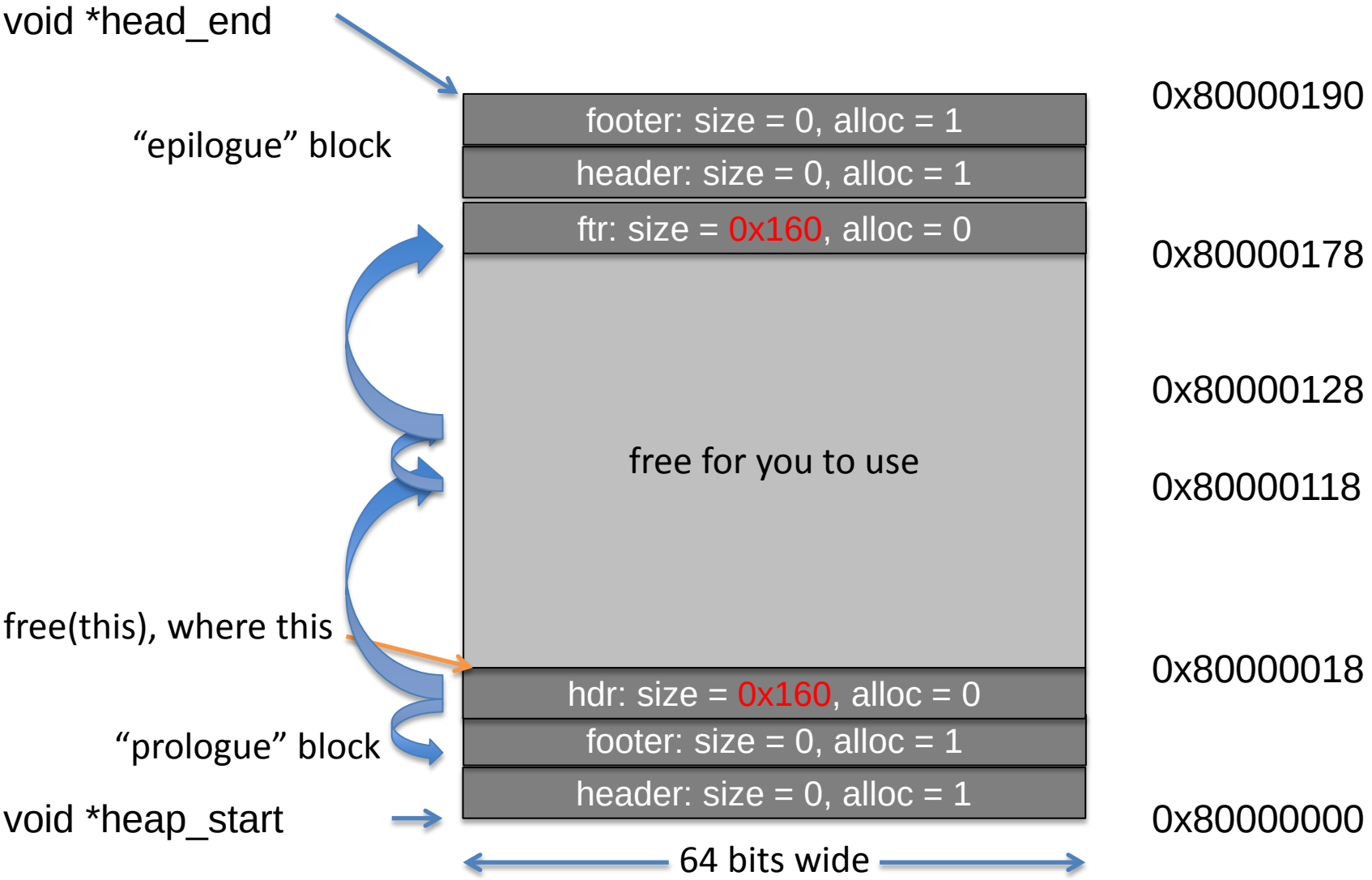
suddenly,free(0x80000018)



suddenly,free(0x80000018)



suddenly,free(0x80000018)



Outline

- ~~Overview~~
- ~~Naïve Implicit List Allocator~~
- Getting Started
- Evaluation
- Debugging Tips, Misc

How to Start (suggestion)

- Read the 32 bit implicit list in your textbook and wade through the sea of macros.
  - Don't copy and paste from the CSAPP website (you won't learn as much).
- Implement a naïve 64 bit implicit list
  - you can leave out coalescing, just for now
- Implement mm_checkheap() for this heap pattern.

How to Finish

- Turn an implicit list into an explicit list
  - update mm_checkheap(), debug, tune
- Turn your explicit list into segregated free lists
  - update mm_checkheap(), debug, tune

Outline

- ~~Overview~~
- ~~Naïve Implicit List Allocator~~
- ~~Getting Started~~
- Evaluation
- Debugging Tips, Misc

```
[student@makoshark]$ ./mdriver
Using default tracefiles in ./traces/
Measuring performance with a cycle counter.
Processor clock rate ~= 2260.7 MHz
...........................
Results for mm malloc:
  valid  util   ops    secs     Kops  trace
    yes    89%  100000  0.004959 20165 ./traces/alaska.rep
  * yes    75%   11991  0.001168 10266 ./traces/chrome.rep
  u yes    59%      --       --    -- ./traces/exhaust.rep
  * yes    77%     200  0.000015 13613 ./traces/lrucd.rep
  u yes    99%      --       --    -- ./traces/needle.rep
  p yes     --    6495  0.004902  1325 ./traces/seglist.rep
    yes    99%      12  0.000003  4105 ./traces/short2.rep
17 16      85%  149036  0.024727  6027

Perf index = 52 (util) & 11 (thru) = 63/100
```

```
if(size == 213) /* large_trace.rep */
{
    // skip searching through free
    // list, because there isn't any
    // block that fits – I checked
    // the trace.
    extend_heap(size);
}
```

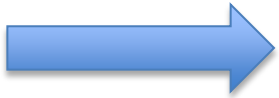
Not a general purpose allocator.

You will be penalized for this.

Outline

- ~~Overview~~
- ~~Naïve Implicit List Allocator~~
- ~~Getting Started~~
- ~~Evaluation~~
- Debugging Tips, Misc

Misc

*p + 0xff | 0x08  SIZE(HD(ptr))

?????

oh.

Misc

#define MAX(x, y) x > y ? x : y

- MAX(x++,y) -> x++ > y ? x++ : y
  - Not what you'd expect

```
static inline int max(x, y) {
   return x > y ? x : y;
}
```

- max(x++, y)?

#define DOUBLE(x) 2 * x

- DOUBLE(x+1) -> 2 * x + 1 (wrong)

#define DOUBLE(x) 2 * (x)

- struct __attribute__ ((__packed__))
  - neat, but gcc specific (not standard)

Debugging Tips, Misc

1. write mm_checkheap().
2. write mm_checkheap() well.
3. write coalescing to make bugs more apparent, fix bugs using mm_checkheap().
4. Start now.
5. Accelerate neutrinos past the speed of light, enabling you to start three days ago.
6. Good luck!