CS107, Lecture 15
Managing The Heap

Reading: B&O 9.9, 9.11

Learning Goals

- Learn the restrictions, goals and assumptions of a heap allocator
- Understand the conflicting goals of utilization and throughput
- Learn about 3 different ways to implement a heap allocator

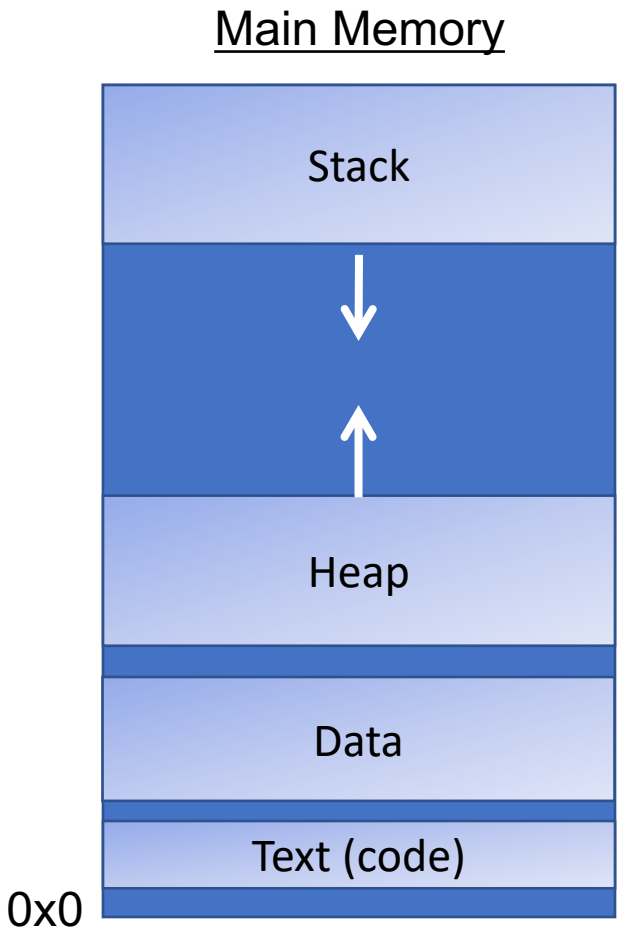
Plan For Today

- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

Plan For Today

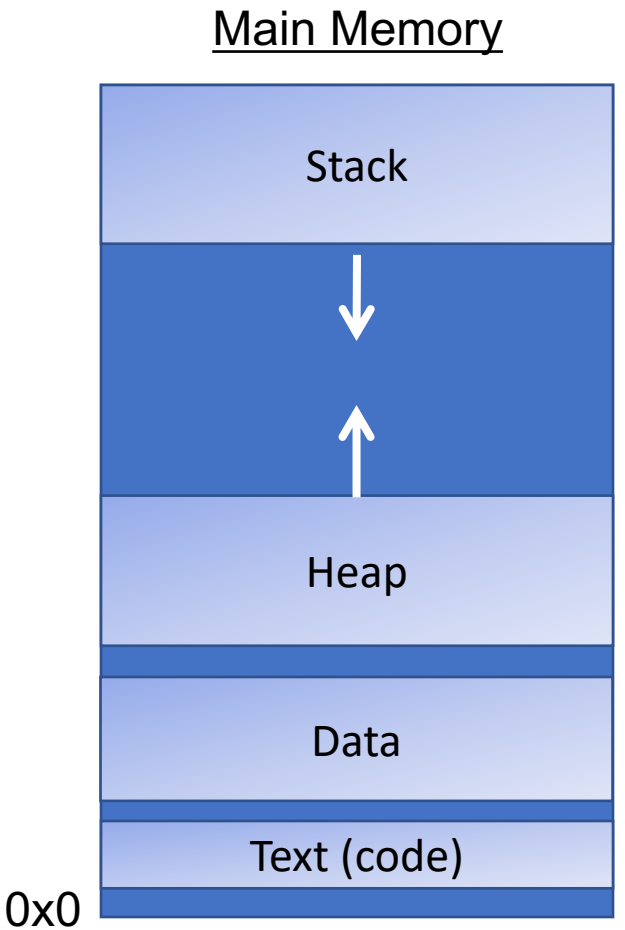
- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

The Heap



The Heap

- The heap is an area of memory below the stack that grows up towards higher addresses.
- Unlike the stack, where memory goes away when a function finishes, the heap provides memory that persists until the caller is done with it.



Plan For Today

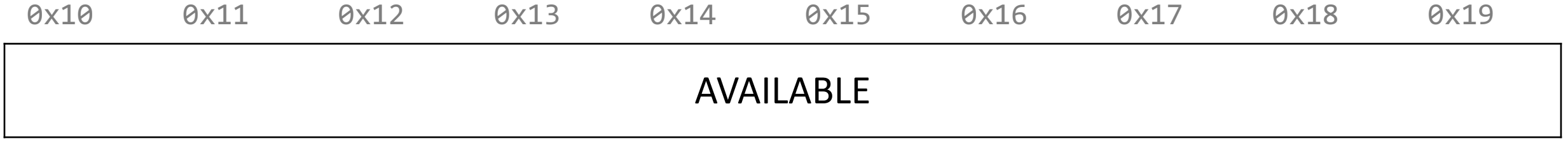
- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.

What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).

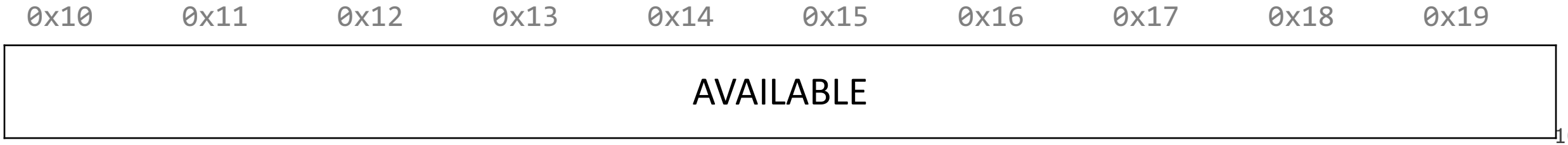


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: Hi! May I please have 2 bytes of heap memory?

Allocator: Sure, I've given you address 0x10.

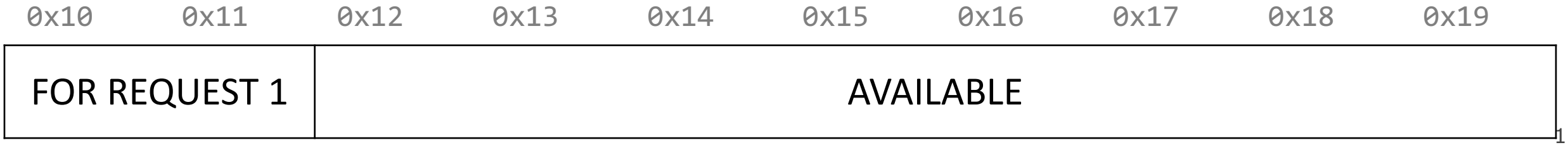


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: Hi! May I please have 2 bytes of heap memory?

Allocator: Sure, I've given you address 0x10.

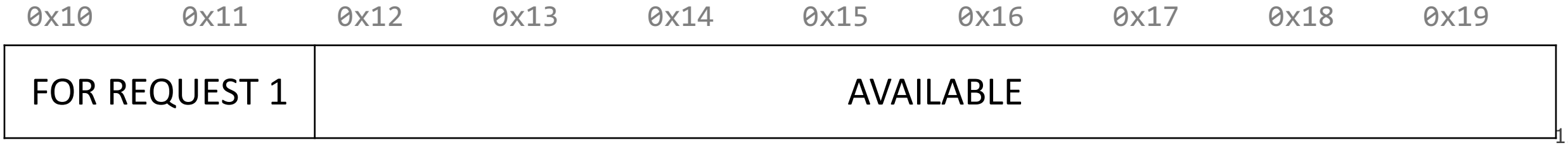


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 2: Howdy! May I please have 3 bytes of heap memory?

Allocator: Sure, I've given you address 0x12.

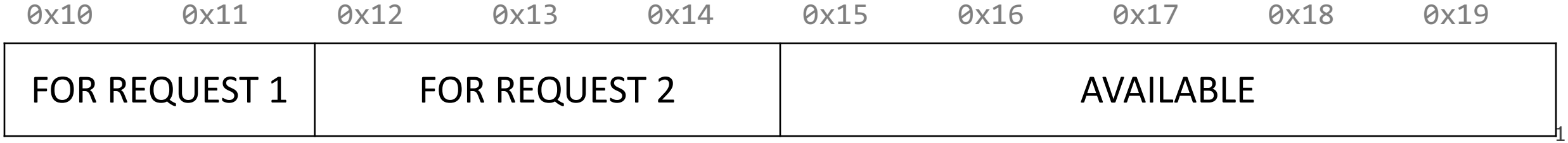


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 2: Howdy! May I please have 3 bytes of heap memory?

Allocator: Sure, I've given you address 0x12.

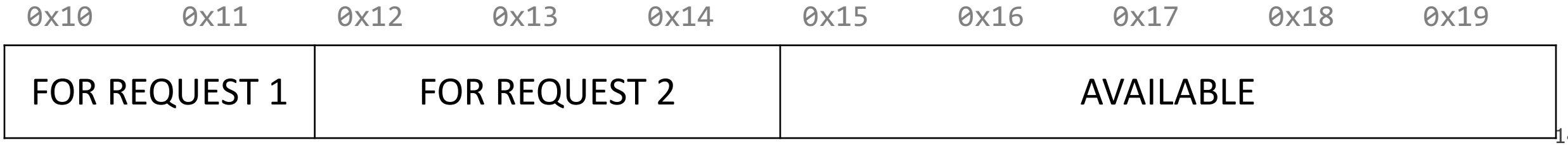


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: I'm done with the memory I requested. Thank you!

Allocator: Thanks. Have a good day!

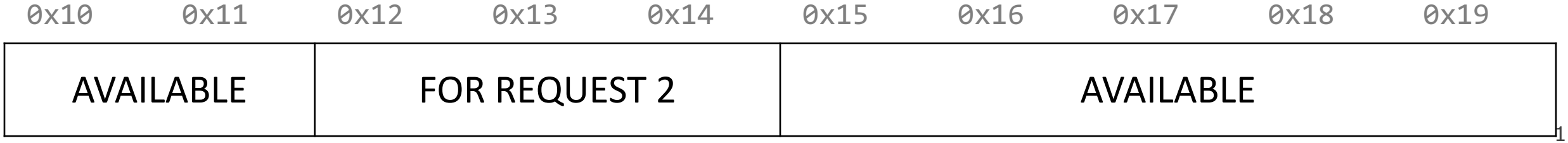


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: I'm done with the memory I requested. Thank you!

Allocator: Thanks. Have a good day!

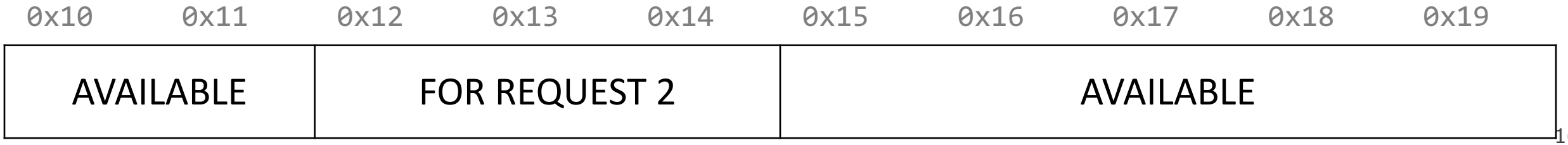


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hello there! I'd like to request 2 bytes of heap memory, please.

Allocator: Sure thing. I've given you address 0x10.

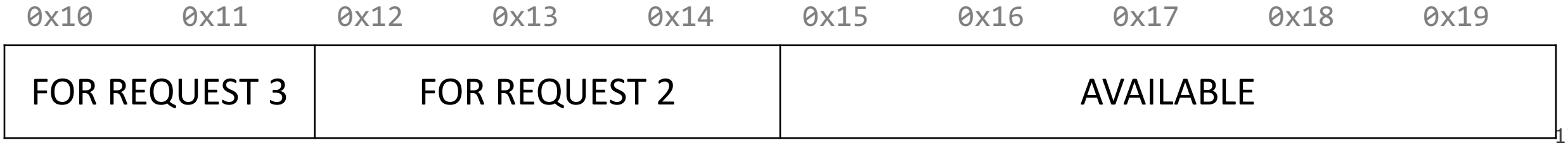


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hello there! I'd like to request 2 bytes of heap memory, please.

Allocator: Sure thing. I've given you address 0x10.

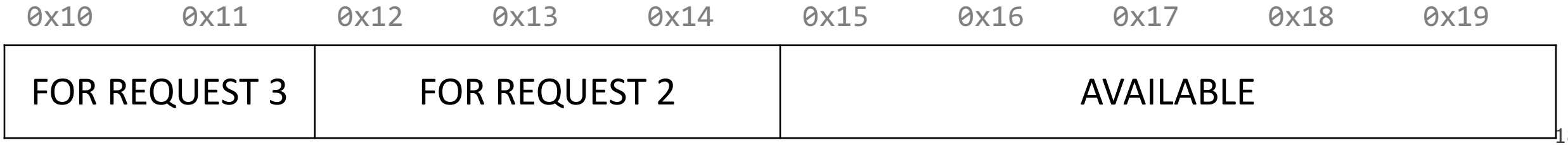


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hi again! I'd like to request the region of memory at 0x10 be reallocated to 4 bytes.

Allocator: Sure thing. I've given you address 0x15.

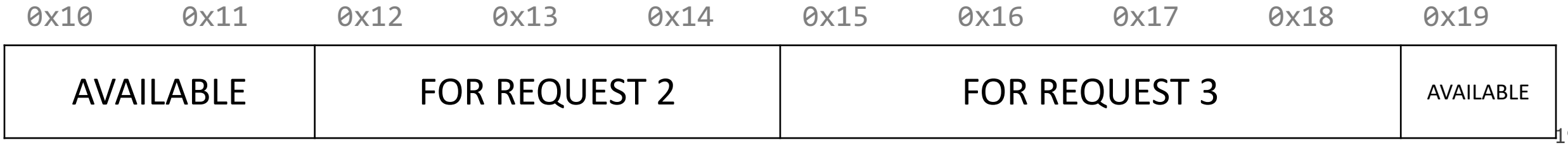


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hi again! I'd like to request the region of memory at 0x10 be reallocated to 4 bytes.

Allocator: Sure thing. I've given you address 0x15.



Plan For Today

- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

You Are the "Heap Hotel Concierge"

Heap Allocator Functions

```
void *malloc(size_t size);
```

Returns a pointer to a block of heap memory of at least size bytes, or NULL if an error occurred.

```
void free(void *ptr);
```

Frees the heap-allocated block starting at the specified address.

```
void *realloc(void *ptr, size_t size);
```

Changes the size of the heap-allocated block starting at the specified address to be the new specified size. Returns the address of the new, larger allocated memory region.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay

Heap Allocator Requirements

A heap allocator must…

1. **Handle arbitrary request sequences of allocations and frees**
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay

A heap allocator cannot assume anything about the order of allocation and free requests, or even that every allocation request is accompanied by a matching free request.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. **Keep track of which memory is allocated and which is available**
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay

A heap allocator marks memory regions as **allocated** or **available**. It must remember which is which so as to properly provide memory to clients.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. **Decide which memory to provide to fulfill an allocation request**
4. Immediately respond to requests without delay

A heap allocator may have options for which memory to use to fulfill an allocation request. It must decide this based on a variety of factors.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. **Immediately respond to requests without delay**

A heap allocator must respond immediately to allocation requests, and should not e.g. prioritize or reorder certain requests to improve performance.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay
5. **Return addresses that are 8-byte-aligned (must be multiples of 8).**

Heap Allocator Goals

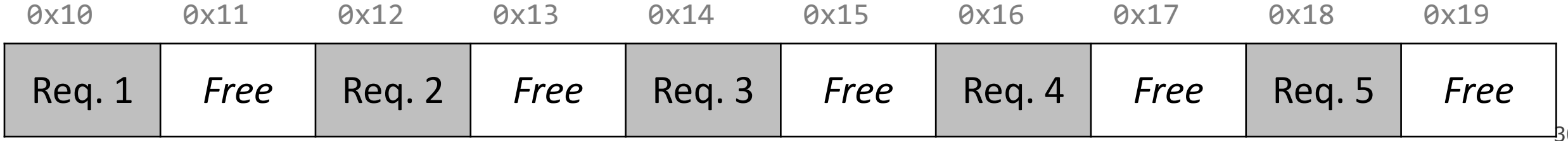
- Goal 1: Maximize **throughput**, or the number of requests completed per unit time. This means minimizing the average time to satisfy a request.
- Goal 2: Maximize memory **utilization**, or how efficiently we make use of the limited heap memory to satisfy requests.

Utilization

- The primary cause of poor utilization is **fragmentation**. **Fragmentation** occurs when otherwise unused memory is not available to satisfy allocation requests.
- In this example, there is enough aggregate free memory to satisfy the request, but no single free block s large enough to handle the request.
- In general: we want the largest address used to be as low as possible.

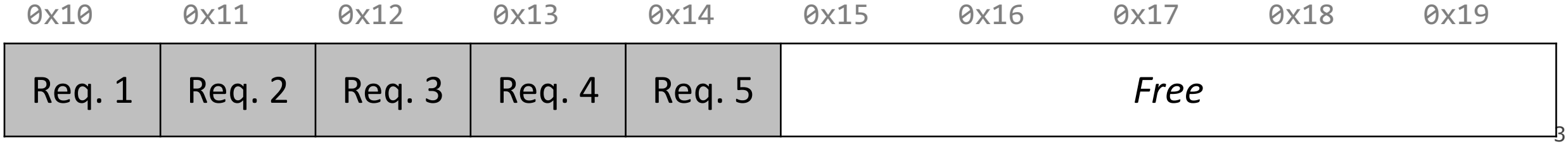
Request 6: Hi! May I please have 4 bytes of heap memory?

Allocator: I'm sorry, I don't have a 4 byte block available…



Utilization

- Question: what if we shifted these blocks down to make more space? Can we do this?
- **No!** We have already guaranteed these addresses to the client. We cannot move allocated memory around, since this will mean the client will now have incorrect pointers to their memory!



Heap Allocator Goals

- Goal 1: Maximize **throughput**, or the number of requests completed per unit time. This means minimizing the average time to satisfy a request.
- Goal 2: Maximize memory **utilization**, or how efficiently we make use of the limited heap memory to satisfy requests.

These are seemingly conflicting goals – for instance, it may take longer to better plan out heap memory use for each request.

Heap allocators must find an appropriate balance between these two goals!

Plan For Today

- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

Bump Allocator

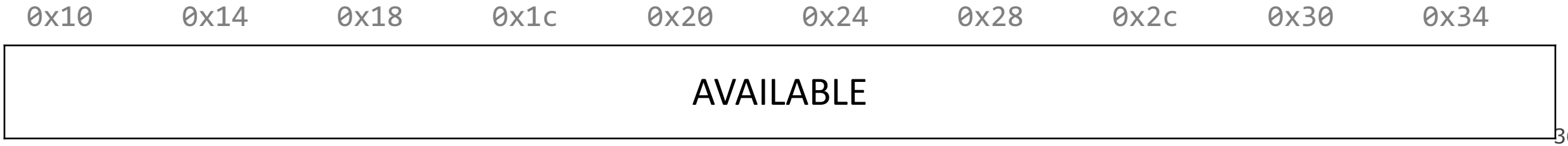
- Let's approach designing our first heap allocator implementation.
- Let's say we want to prioritize throughput as much as possible, and do not care about utilization at all. How could we do this?
- Discuss! (*Hint: does **free()** really need to do anything?*)

Bump Allocator

- A **bump allocator** is a heap allocator design that simply allocates the next available memory address upon an allocate request, and does nothing on a free request.
- Throughput: each **malloc** and **free** execute only a handful of instructions:
  - It is easy to find the next location to use
  - Free does nothing!
- Utilization: we use each memory block at most once. No freeing at all, so no memory is ever reused. ☹
- We provide a bump allocator implementation as part of assign7 as a code reading exercise.

Bump Allocator

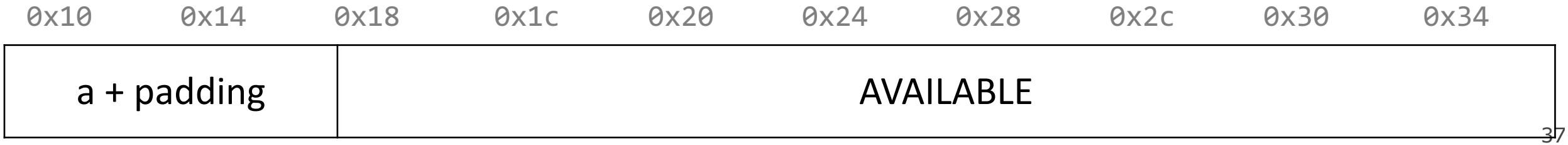
```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```



Bump Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```

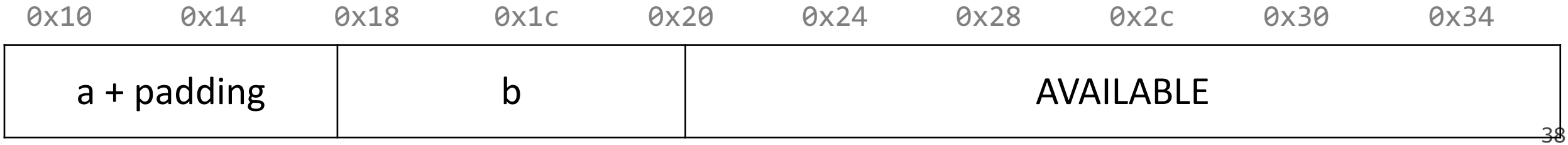
Variable	Value
a	0x10



Bump Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```

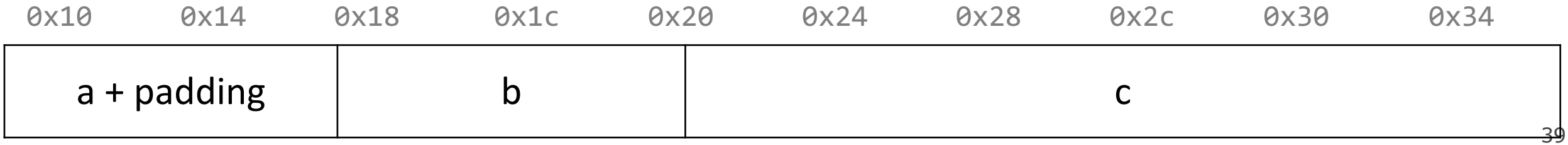
Variable	Value
a	0x10
b	0x18



Bump Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```

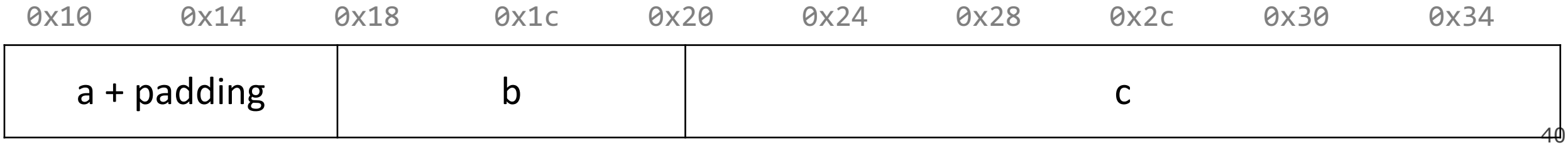
Variable	Value
a	0x10
b	0x18
c	0x20



Bump Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```

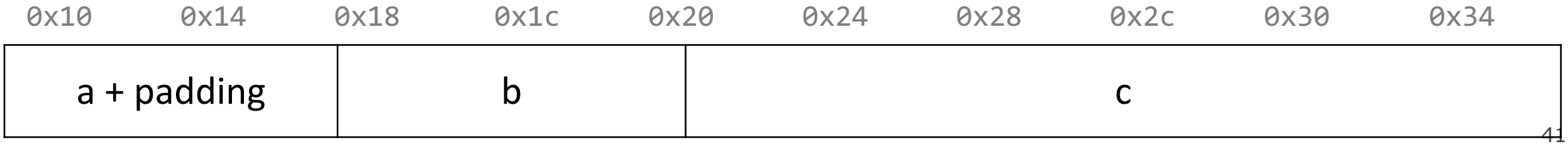
Variable	Value
a	0x10
b	0x18
c	0x20



Bump Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(24);
free(b);
void *d = malloc(4);
```

Variable	Value
a	0x10
b	0x18
c	0x20
d	NULL



Bump Allocator

- A bump allocator is an extreme heap allocator – it optimizes only for **throughput**, not **utilization**.
- Better allocators strike a more reasonable balance. How can we do this?

Questions to consider:

1. How do we keep track of free blocks?
2. How do we choose an appropriate free block in which to place a newly allocated block?
3. After we place a newly allocated block in some free block, what do we do with the remainder of the free block?
4. What do we do with a block that has just been freed?

Announcements

- Last lab (**lab8**) this week on optimization and efficiency. Useful to fine tune your heap allocator!
- Heap allocator assignment goes out tonight, due **Thurs. 3/14 at 11:59PM PST**. Grace period until **Fri. 3/15 at 11:59PM PST.**
- Midterm regrade requests will be processed over the next few days.

Can we do better?

1. How do we keep track of free blocks?
2. How do we choose an appropriate free block in which to place a newly allocated block?
3. After we place a newly allocated block in some free block, what do we do with the remainder of the free block?
4. What do we do with a block that has just been freed?

Plan For Today

- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

Implicit Free List Allocator

- **Key idea:** in order to reuse blocks, we need a way to track which blocks are allocated and which are free.
- We could store this information in a separate global data structure, but this is inefficient.
- Instead: let's allocate extra space before each block for a **header** storing its size and whether it is allocated or free.
- When we allocate a block, we look through the blocks to find a free one, and we update its header to reflect its allocated size and that it is now allocated.
- When we free a block, we update its header to reflect it is now free.
- The header can be 8 bytes to make alignment easier.

Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

Implicit Free List Allocator

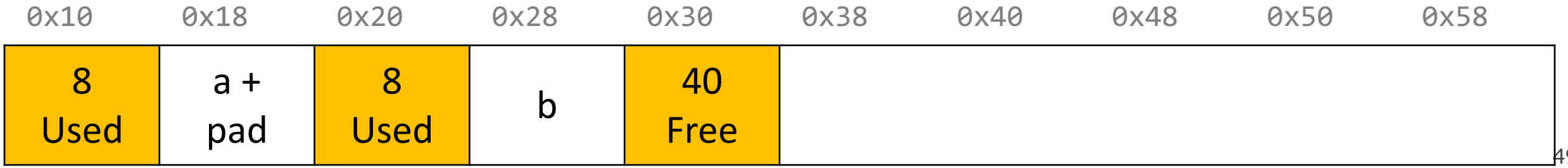
```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

Variable	Value
a	0x18

Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

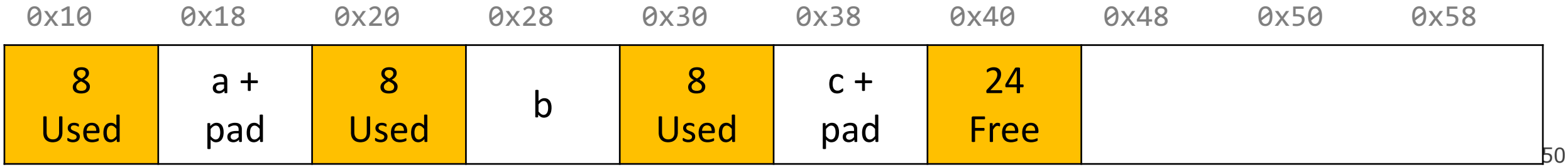
Variable	Value
a	0x18
b	0x28



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

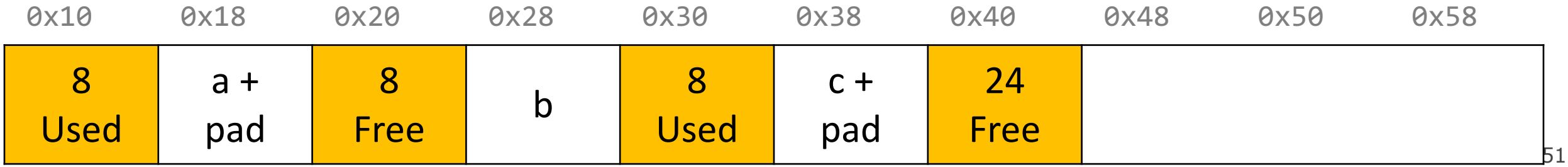
Variable	Value
a	0x18
b	0x28
c	0x38



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

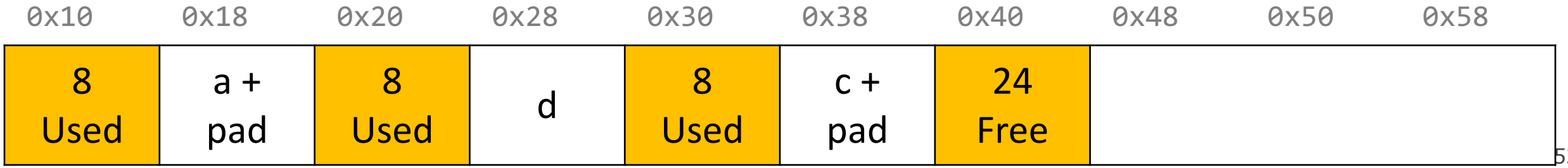
Variable	Value
a	0x18
b	0x28
c	0x38



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

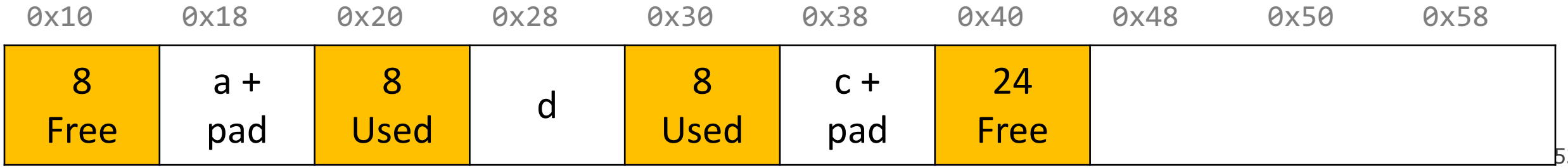
Variable	Value
a	0x18
b	0x28
c	0x38
d	0x28



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

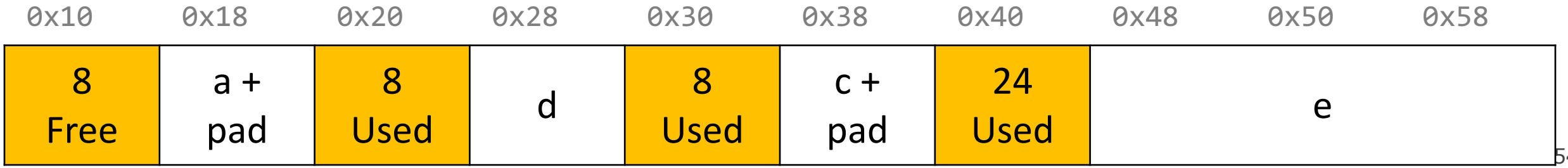
Variable	Value
a	0x18
b	0x28
c	0x38
d	0x28



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

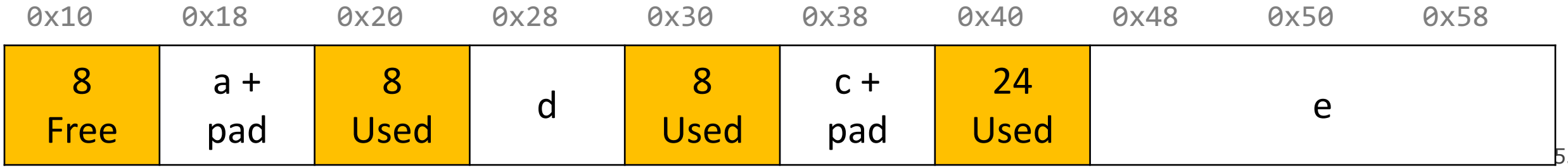
Variable	Value
a	0x18
b	0x28
c	0x38
d	0x28
e	0x48



Implicit Free List Allocator

```
void *a = malloc(4);
void *b = malloc(8);
void *c = malloc(4);
free(b);
void *d = malloc(8);
free(a);
void *e = malloc(24);
```

Variable	Value
a	0x18
b	0x28
c	0x38
d	0x28
e	0x48



Implicit Free List Allocator

- How can we choose a free block to use for an allocation request?
  - **First fit:** search the list from beginning each time and choose first free block that fits.
  - **Next fit:** instead of starting at the beginning, continue where previous search left off.
  - **Best fit:** examine every free block and choose the one with the smallest size that fits.

- What are the pros/cons of this approach?

Plan For Today

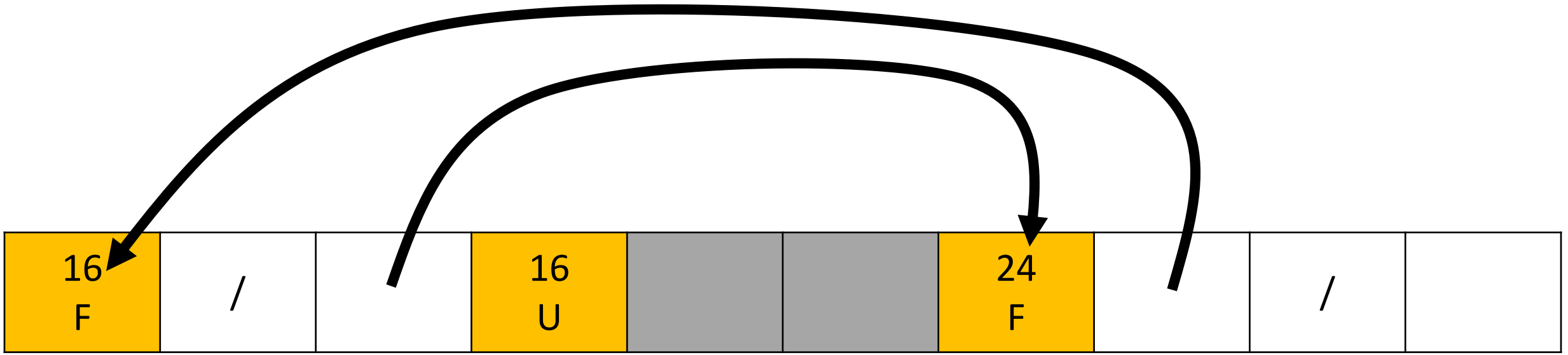
- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

Explicit Free List Allocator

- **Key idea:** an implicit free list requires us to iterate over every block (allocated and free) to find a free block to use for an allocation.
- Is it possible for us to be able to skip over allocated blocks and just hop between *free blocks*?

Explicit Free List Allocator

Key Idea: free blocks have payloads that are unused. Let's *use that space to store pointers to the next and previous free block!*



Recap

- Recap: the heap
- What is a heap allocator?
- Heap allocator requirements and goals
- Method 1: Bump Allocator
- Method 2: Implicit Free List Allocator
- Method 3: Explicit Free List Allocator

Next time: more heap allocators, and optimizing efficiency