CS 354 - Machine Organization & Programming
Tuesday, October 8, 2019

Last Time

C's Heap Allocator (`stdlib.h`)
Posix `brk` (`unistd.h`)
Allocator Design

Today

Simple View of Heap
Free Block Organization
Implicit Free List
Placement Policies

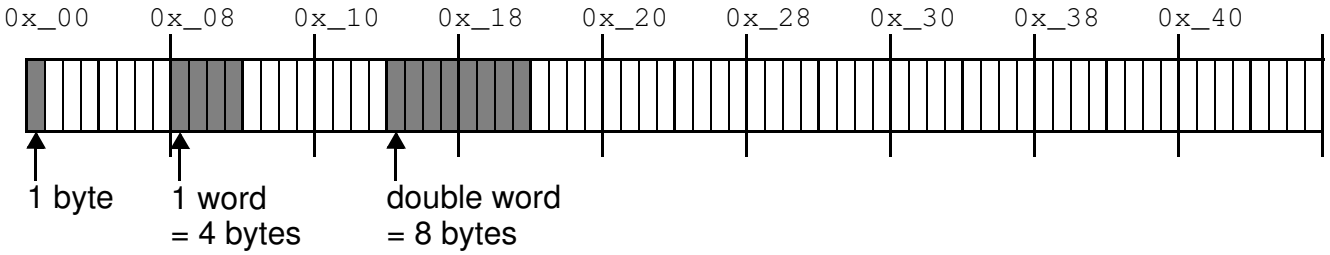
Exams Returned

Next Time

Splitting, Coalescing, Footers, Explicit Free Lists
Read: B&O 9.9.9 - 9.9.11, 9.9.13
Skim: B&O 9.9.12

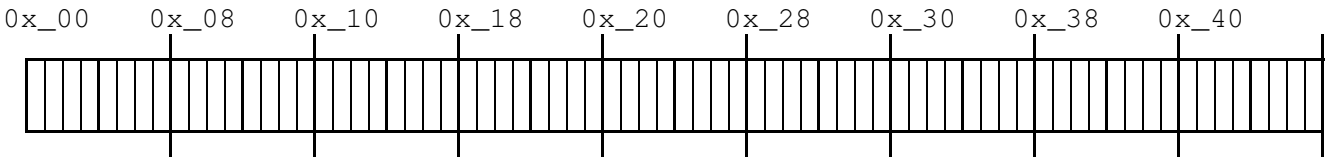
Simple View of Heap

Linear Memory Layout



double word alignment:

Heap Allocation Run 1 with a Simple View



→ Update the diagram to show the following heap allocations:

```
1) p1 = malloc(2 * sizeof(int));
2) p2 = malloc(3 * sizeof(char));
3) p3 = malloc(4 * sizeof(int));
4) p4 = malloc(5 * sizeof(int));
```

→ What happens with the following heap operations:

```
5) free(p1); p1 = NULL;
6) free(p3); p3 = NULL;
7) p5 = malloc(6 * sizeof(int));
```

External Fragmentation:

Internal Fragmentation:

Free Block Organization

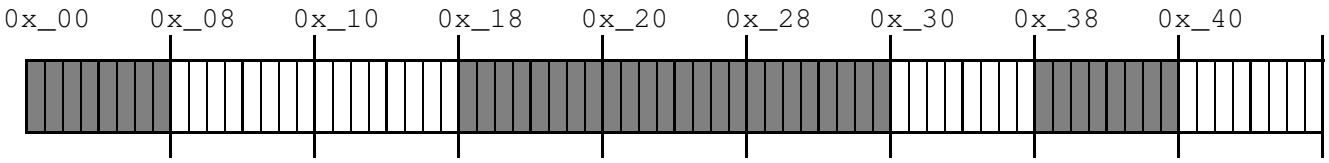
❋ *Simple view of allocator has*

size

status

Explicit Free List

◆



code:

space:

time:

Implicit Free List

◆

code:

space:

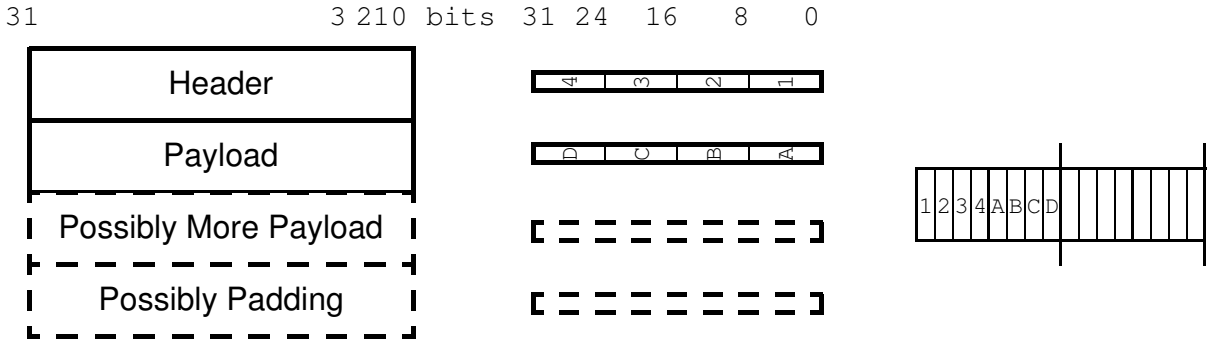
time:

Implicit Free List

❋ *The first word of each block is*

→ Since the block size is a multiple of 8, what value will the last three header bits always have?

Basic Heap Block Layout

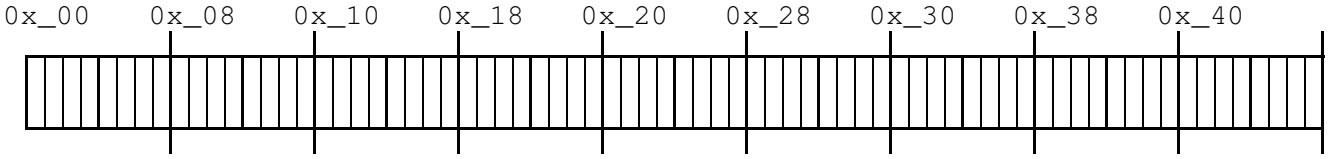


→ What integer value will the header have for a block that is

1) allocated and 8 bytes in size?
2) free and 32 bytes in size?
3) allocated and 64 bytes in size?

❋ *The header is an integer*

Heap Allocation Run 2 with Block Headers



→ Update the diagram to show the following heap allocations:

```
1) p1 = malloc(2 * sizeof(int));
2) p2 = malloc(3 * sizeof(char));
3) p3 = malloc(4 * sizeof(int));
4) p4 = malloc(5 * sizeof(int));
```

➢ Why does it make sense that Java doesn't allow primitives on the heap?

Placement Policies

What? *Placement Policies* are

Assume the heap is pre-divided into various-sized free blocks ordered from smaller to larger.

- First Fit (FF): start from
  stop at
  fail if

  mem util:

  thruput:

- Next Fit (NF): start from
  stop at
  fail if

  mem util:

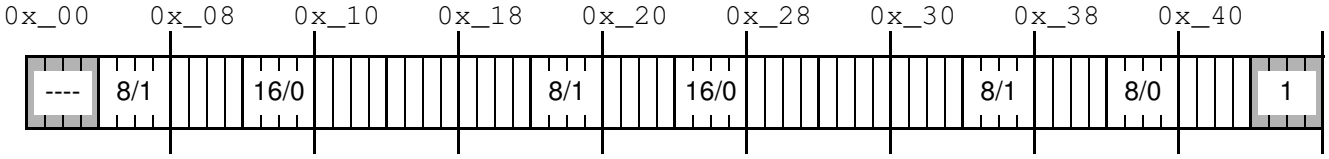
  thruput:

- Best Fit (BF): start from
  stop at
  or stop early
  fail if

  mem util:

  thruput:

Heap Allocation Run 3 using a Placement Policy



→ Given the original heap above and the placement policy, what address is `ptr` assigned?

```
ptr = malloc(sizeof(int));          //FF?              BF?

ptr = malloc(10 * sizeof(char));    //FF?              BF?
```

→ Given the original heap above and the address of block most recently allocated, what address is `ptr` assigned using NF?

```
ptr = malloc(sizeof(char));         //0x_04?             0x_34?

ptr = malloc(3 * sizeof(int));      //0x_1C?             0x_34?
```

→ Given a pointer to the first block in the heap, how is the next block found?