Memory Allocation I

CSE 351 Autumn 2023

Instructor:
Justin Hsia

Teaching Assistants:

Afifah Kashif	Malak Zaki
Bhavik Soni	Naama Amiel
Cassandra Lam	Nayha Auradkar
Connie Chen	Nikolas McNamee
David Dai	Pedro Amarante
Dawit Hailu	Renee Ruan
Ellis Haker	Simran Bagaria
Eyoel Gebre	Will Robertson
Joshua Tan	

Adapted from
https://xkcd.com/1093/

Relevant Course Information

- hw17 due tonight
- hw19 due Friday (11/17)
  - Lab 4 preparation!
- hw20 due Monday (11/20)

- Lab 4 due Monday after Thanksgiving (11/27)
  - Section tomorrow intended to help prepare you for Lab 4

- Midterm scores posted
  - See Ed post #966 for common misconceptions and deductions
  - Regrade requests open from Nov. 16-18 (Thu-Sat)

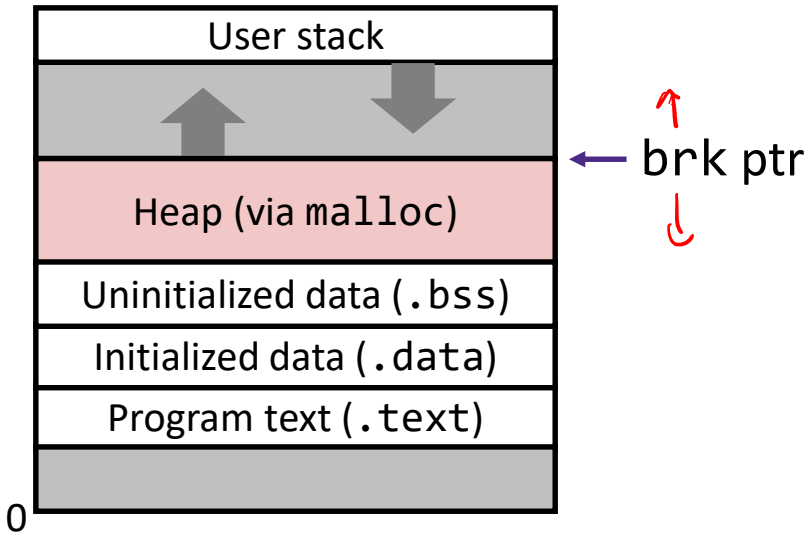
Growth vs. Fixed Mindset

- Students can be thought of as having either a “growth” mindset or a “fixed” mindset (based on research by Prof. Carol Dweck)
  - “In a **fixed mindset** students believe their basic abilities, their intelligence, their talents, are just fixed traits. They have a certain amount and that's that, and then their goal becomes to look smart all the time and never look dumb.”
  - “In a **growth mindset** students understand that their talents and abilities can be developed through effort, good teaching and persistence. They don't necessarily think everyone's the same or anyone can be Einstein, but they believe everyone can get smarter if they work at it.”

Memory Allocation I

Lesson Summary (1/3)

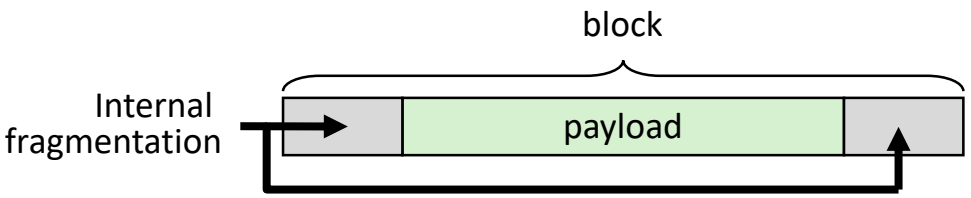
- **Dynamic memory allocation** is used when size or lifetime is not known until runtime
  - Memory allocated in the heap segment of memory: (dynamically sized)
  - In C: `void* malloc(size_t size)` (NULL on error)
  - In C: `void free(void* p)`
  - In Java: **`new`**

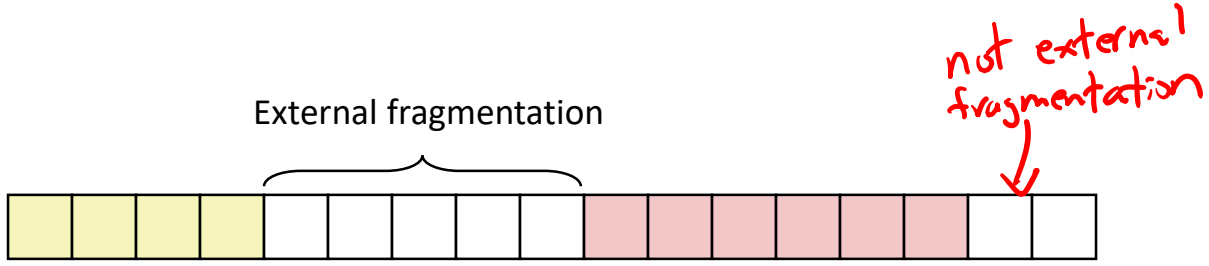


- Managed by dynamic memory allocator
  - Implicit: automatic deallocations (Java garbage collection), Explicit: manual deallocations (C)
  - Performance metrics: throughput, memory utilization ← ratio of payload to heap size
    ↳ how quickly can I allocate & deallocate

Lesson Summary (2/3)

- The heap is divided into allocated and free **heap blocks**
  - Fragmentation: internal is non-payload space within blocks, external is free space between allocated blocks

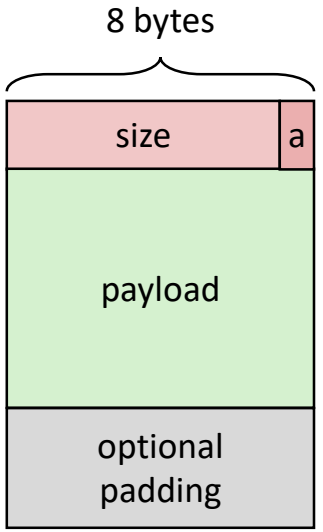




  - Blocks have headers with size and is-allocated? Information:

```
x = size | a;
a = x & 1;
size = x & ~1;
```

Format of allocated and free blocks:



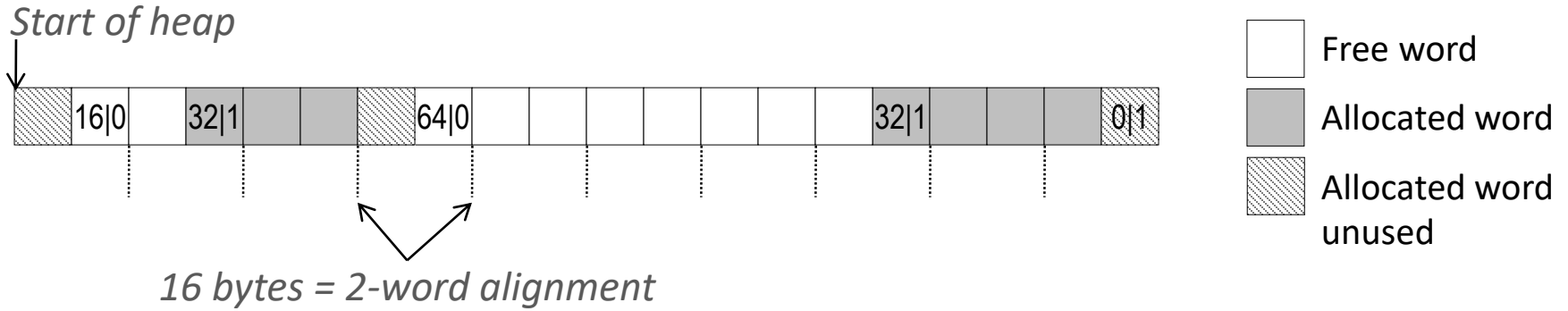
a = 1: allocated block
a = 0: free block

size: block size (in bytes)

payload: application data (allocated blocks only)

Lesson Summary (3/3)

- Implicit free list example
- Heap blocks (`size`|is-allocated?): 16|0, 32|1, 64|0, 32|1



- 16-byte alignment for (1) *heap block size* and (2) *payload address*
  - Padding for size is considered part of *previous* heap block (internal fragmentation)
  - May require initial padding at start of heap
- Special one-word marker (0|1) marks end of list
  - Zero `size` is distinguishable from all other blocks

Lesson Q&A

- Terminology:
  - Dynamically-allocated data: malloc, free
  - Allocators: implicit vs. explicit allocators, heap blocks, implicit vs. explicit free lists
  - Heap fragmentation: internal vs. external, padding, alignment
  - Header, heap block size, is-allocated? bit

- Learning Objectives:
  - Use `malloc` and `free` in C programs to manage dynamic data.
  - Explain the tradeoffs between different allocator implementations, [policies, and strategies].

- What lingering questions do you have from the lesson?

Memory Allocation I – Practice

Practice Questions (1/2)

- Which of the following statements is FALSE?

A. Temporary arrays should *not* be allocated on the Heap

should allocate on the Stack

B. `malloc` returns an address of a payload that is filled with mystery data

allocates only; no initialization

C. Peak memory utilization is a measure of both internal and external fragmentation

$\frac{\text{aggregate payload}}{\text{heap size}}$

D. An allocation failure will cause your program to stop

just returns NULL

E. We're lost…

Practice Questions (2/2)

- How many "flags" can we fit in our header if our allocator uses 16-byte alignment?

all multiples of 16 have lowest 4 bits as zeros. ⇒ 16 = 0b10000

4 flags

- If we placed a new "flag" in the second least significant bit, write out a C expression that will extract this new flag from header

two steps: ① mask out bit ②
② shift into LSB ①

(header & 2) >> 1

(header >> 1) & 1

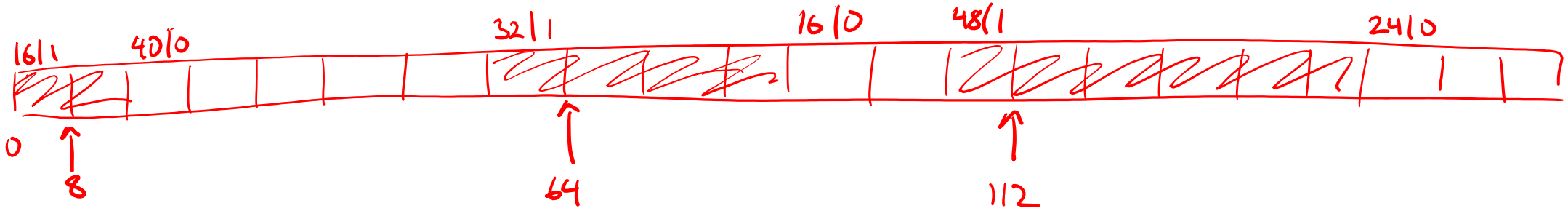
Homework Setup

Free word

Allocated word

- *Implicit free list* that uses **8-byte headers** and **8-byte alignment**. The current blocks on the heap are numbered and sized as follows.
  - Draw out the heap starting at address "0". What is the address of each allocated block's payload? 8, 64, 112
  - How much padding does each allocated block have? padding = size - header - payload

#	Size	Type	Request	
1	16 B	allocated	`malloc(8)`	16-8-8 = 0 B
2	40 B	free	n/a	
3	32 B	allocated	`malloc(20)`	32-8-20 = 4 B
4	16 B	free	n/a	
5	48 B	allocated	`malloc(35)`	48-8-35 = 5 B
6	24 B	free	n/a	



Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the homework problems
  3) Work on the current lab

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support