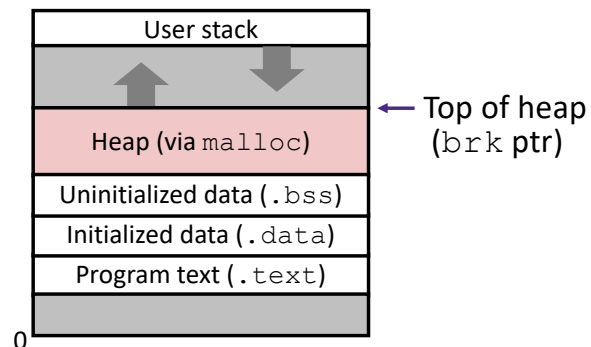


Dynamic Memory Allocation

- ❖ Allocator organizes heap as a collection of variable-sized *blocks*, which are either *allocated* or *free*
 - Allocator requests pages in the heap region; virtual memory hardware and OS kernel allocate these pages to the process
 - Application objects are typically smaller than pages, so the allocator manages heap blocks *within* pages
 - (Larger objects handled too; ignored here)

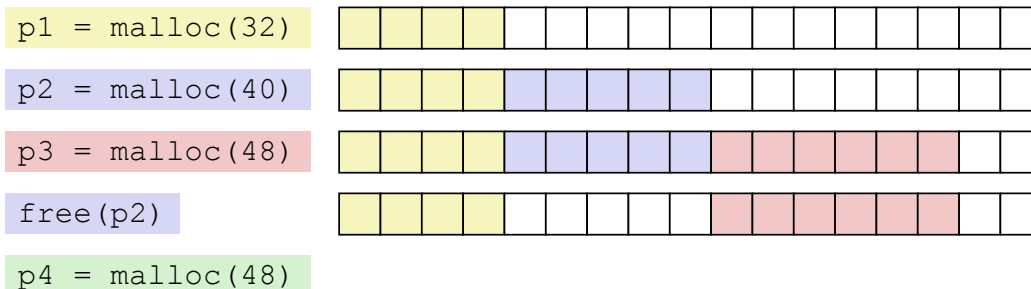


43

External Fragmentation

= 8-byte word

- ❖ For the heap, *external fragmentation* occurs when allocation/free pattern leaves “holes” between blocks
 - That is, the aggregate payload is non-continuous
 - Can cause situations where there is enough aggregate heap memory to satisfy request, but no single free block is large enough



44

Header Questions

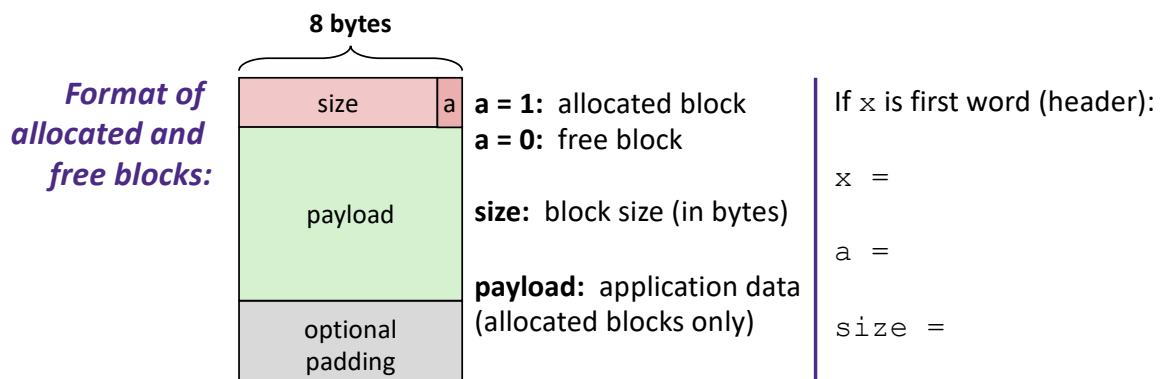
- ❖ How many “flags” can we fit in our header if our allocator uses 16-byte alignment?
- ❖ If we placed a new “flag” in the second least significant bit, write out a C expression that will extract this new flag from `header`

45

Implicit Free Lists (Handout)

- ❖ For each block we need: **size, is-allocated?**
 - Could store using two words, but wasteful
- ❖ Standard trick
 - If blocks are aligned, some low-order bits of `size` are always 0
 - Use lowest bit as an allocated/free flag (fine as long as aligning to $K > 1$)
 - When reading `size`, must remember to mask out this bit!

e.g., with 8-byte alignment,
possible values for size:
00001000 = 8 bytes
00010000 = 16 bytes
00011000 = 24 bytes
...



46