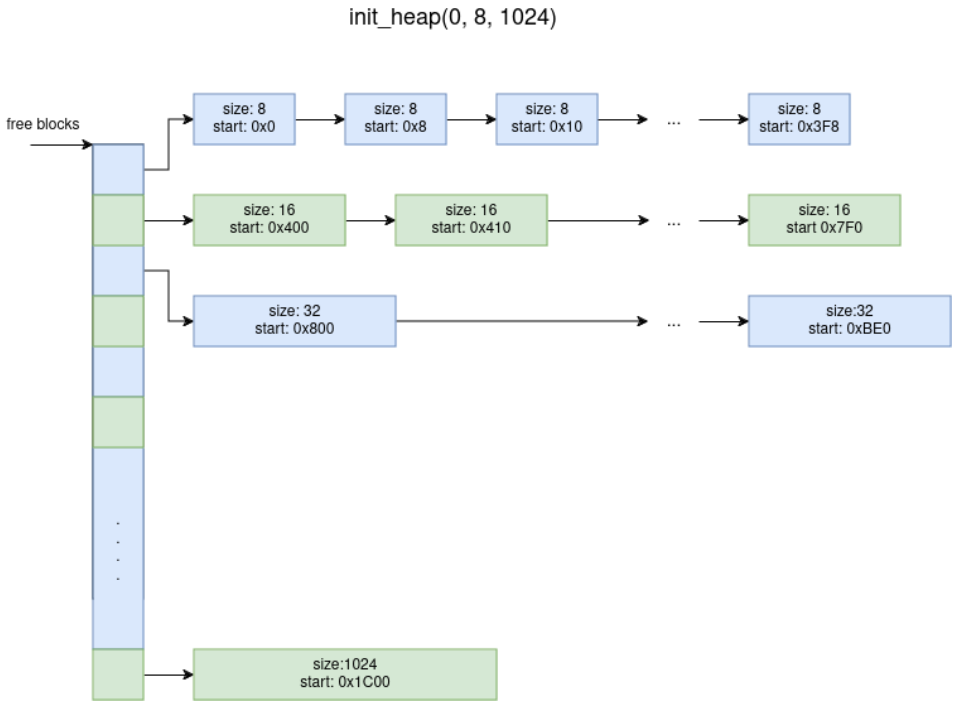
Definition

Segregated Free Lists represent a data structure consisting of a vector where each entry stores the starting address of a doubly linked list containing free memory blocks of the same size. Initially, the sizes of the blocks are powers of 2, so that after allocations, they can be fragmented or merged as necessary. As the name suggests, this data structure holds only unallocated blocks, with a separate data structure being used for allocated blocks.



Requirement

The memory allocator's role is to reserve memory at the library level using traditional calls like malloc() or calloc(). These calls mark certain memory areas from a pre-allocated pool as used, which in this task is organized in the form of segregated free lists. The memory allocator also manages the release of reserved areas, with the corresponding library call being free().

The memory allocator I have implemented has the following functionalities:

- A vector of doubly linked lists called segregated free lists, where each entry points to a doubly linked list holding free memory areas of the same size.

- A doubly linked list which manages the allocated memory area blocks, along with their content.

Possible Commands

Input is provided from stdin, and output is written to stdout in the following format:

• INIT_HEAP <start_heap_address> <number_of_lists> <number_of_bytes_per_list> <reconstruction_type>

• MALLOC <number_of_bytes>

• FREE <address>

• WRITE <address> <data> <number_of_bytes>

• READ <address> <number_of_bytes>

Note: It is guaranteed that the READ operation will always read from areas previously written to with WRITE. In other words, a valid READ operation will always follow a valid WRITE operation.

• DUMP_MEMORY

• DESTROY_HEAP

Bonus

As a bonus, if the <reconstruction type> parameter in the INIT_HEAP command is set to 1, which means reconstructing blocks during FREE that were fragmented from a larger one. Reconstruction also applies to intermediate fragments, where a large block was fragmented, and one of its fragments was further divided, and so on. In this case, the reconstruction should be recursive.

Example: Suppose we have a single free block of size 16 bytes starting at address 0x1.

MALLOC 6 -> The 16-byte block is split into a 6-byte part, which is allocated, and a 10-byte part, which remains free. The address 0x1 is returned.

MALLOC 5 -> The 10-byte block is split into a 5-byte part, which is allocated, and a 5-byte part, which remains free. The address 0x7 is returned.

FREE 0x7 -> The 5-byte fragment is merged with the free 5-byte fragment, resulting in a 10-byte block.

FREE 0x1 -> The 6-byte fragment is merged with the free 10-byte fragment, resulting in a 16-byte block.