CS 241, Lecture 21: Optimization Continued and Heap Management

1 Optimization

Covered last time!

2 Heap Management

- How can we manage heap?
- One way is to just set up a pointer to the beginning of memory, and another to the end. Initalization is O(1), as is allocation, and we never delete... but we get a ton of waste and we will run out of memory!
- Another way is to use linked list of fixed size blocks. This only works because we allow a single block of memory each time, where we keep track of hte memory in the free list and allocated from the free list if memory is requested.
- The next way of doing it is variable sized blocks. We again use a linked list, but here, our linked list will store a number of *bytes*, where those bytes can be found in the next node.
- When we init with this, we start with the entire heap being free. Our linked list has one node, which is the size (1024 bytes, for example) and the starting address (0x4000, for example).
- When we allocate, say, 50 bytes, we allocate *54* bytes - the first 4 bytes are the size of the block (an integer) and the rest is the memory requested in bytes. We return a pointer to the start of the 50 bytes.
- Now, we have 970 bytes and 0x4036 as our address.
- Repeat for 28 bytes. We allocate 32 bytes, and return 0x4056 as our address pointer to the start of the 28 bytes.
- How about freeing? Let's say we free the 50 bytes first. Free chunks and add a linked list node to the free linked list.
- We can merge adjacent memory blocks. Note that 54 + 0x4000 = 0x4036, and so we can collapse the first two nodes in our linked list into a single node.
- We have an issue with fragmentation though.