TPS #1:

1. What is cache? Why do we need cache?
    - It is a hardware/software component that stores memory that is likely to be accessed in the future in order to speed up future computing; We need cache because if we had to directly access RAM for each set of information during a code, it would take way too long to run anything.
2. There are generally 2 practical ways to organize a cache: Direct-mapped cache and N-way set associative cache. In both types of cache, data at a specific address of the main memory (RAM) are mapped to a pre-defined location in the cache. A "Block" is the basic unit of data being mapped between the main memory and cache. The size of a block depends on the specification of a cache. Every time data is transferred between cache and the main memory, it is a block of data being transferred. In this exercise, we will explore the Direct-mapped cache.
    - Done.
3. In a Direct-mapped cache, the cache is organized as a hash table. Addresses of the main memory are mapped to the indices of the cache (block numbers) using a modulo operator (%) as the hash function. As a result, we can divide a memory address into 3 fields: tag, index, offset.
    - Done.
4. Offset bits tell us how many bytes of data are in a block. These bits are the right-most bits of the memory address. You can consider this as the number of columns of data in a cache. With a specific value of the offset bits from an address, we know which column of a block we are trying to access. Given the block size of a cache is 16B (bytes), how many bits do we need for offset? What is the number of bits in offset as a function of block size? Is it practical to have a cache of block size = 1 byte?
    - We need 8 x 16 = 128 bits for an offset equivalent to 16 bytes. 4 x size in bytes = offset in bits. It wouldn't be practical if the cache block size was equal to 1 byte since the entire point of having cache is to store as much surrounding/relevant information from RAM to a more localized point from which the program can access it.
5. Index bits tell us how many blocks there are in a cache. These bits are the next right-most bits of the memory address after the offset bits. You can consider this as the number of blocks (rows) of data in a cache. With a specific value of the index bits from an address, we know which block (row) we are trying to access. Given there are 64 blocks in a cache, how many index bits do we need? What is the number of bits in the index as a function of number of blocks?
    - We need 6 bits to represent 64 different blocks. # of bits = log2(# of blocks).
6. Once you know the number of blocks and the block size of a cache, do you know the total size of the cache? How?

- Yes, we can solve for the number of bits it will take to represent a block and multiply it by the number of blocks to determine the total number of bits the cache is made up of.

7. Since the size of cache is always smaller than the size of the main memory, the sum of bits of the offset and index of a cache will be less than the number of bits in an address of the main memory. What do we do to the left-over bits from the address? Why are they important?
    - The leftover bits from the address form the tag and are used to identify the block of cache we want to access.
8. Given a memory address of 20 bits (during Intel 8086 era), 128B of direct-mapped cache, and 8B block size, answer the following questions:

    a. How big is this main memory?
    - 2^20

    b. How many offset bits?
    - log2(8) = 3 offset bits

    c. How many blocks are there in the cache?
    - 16 blocks, n*b = 128, b = 8, n = 16

    d. How many index bits?
    - 4 is the number of rows, so 2^4 = 16

    e. How many tag bits?
    - 20 - 4 - 3 = 13 bits for the tag

    f. Draw the layout of the cache: including tags, valid bits, dirty bits, and data blocks.
    - Tag Dirty Valid Index Offset

    - 8 rows deep per block of data for 16 blocks

    g. What is the number of bits per row of the cache (number of bits being used in a row: tag, valid bit, dirty bits, and data block)?
    - 79 bits

TPS #2:

1. What is the disadvantage of a Direct-mapped cache? What kind of cache miss will it introduce?
    - If two blocks map to the same location in the cache, they will be continuously referenced interchangeably and this introduces a cache miss called thrashing.
2. To overcome this problem, we can allow multiple blocks of data to occupy the same set of a cache. Note that we use "set" here instead of index of cache. In this organization, we group N blocks (rows) of cache into a set and allow more than one block of data to stay within a set. The layout of the cache remains the same as its direct-mapped version, but the difference is that every N blocks are now being grouped into a set.
    - Done.
3. The memory address is still partitioned into the same 3 fields, but the index bits now refer to the set number. Given a cache with 1024 blocks and the associativity is 4 (4 blocks per set), how many index bits do we need? What is the number of bits in index as a function of number of blocks and associativity?
    - 7 index bits for 1024 blocks with 4 blocks per set. The formula is log2(# blocks/associativity)
4. Given a memory address of 20 bits (during Intel 8086 era), 128B of 2-way cache, and 8B block size, answer the following questions:

    a. How big is this main memory?
    - 2^20

    b. How many offset bits?
    - 4 offset bits, log2(8) = 3 + 1 because 2-way cache requires another bit to distinguish between which of the blocks we are referring to in each set.

    c. How many blocks are there in the cache?
    - n*b = 128, b = 8, n = 16 * 2 = 32 blocks.

    d. How many sets are there in the cache?
    - 16 sets

    e. How many index bits?
    - log2(16 sets) = 4

    f. How many tag bits?
    - 20 - 4 - 4 = 16

    g. Draw the layout of the cache: including tags, valid bits, dirty bits, and data blocks. Indicate the sets with a different color (or a thicker) border.
    - Tag Dirty Valid Index Offset

- 2nd block of data with the same offset and index bits maps to corresponding row in the second block. When the CPU wants to access data stored at a specific index and offset, it will compare the two different tags to see if one matches; if none match, the cache must fetch the data from RAM and then the CPU may access it.

h. What is the number of bits per row of the cache (number of bits being used in a row: tag, valid bit, dirty bits, and data block)?

- 79 bits