1. Cache principles
    a. Cache holds a much smaller subset of RAM's contents
        i. Bring in values from RAM while executing a program
        ii. If we need a value, look in cache first before going to memory
        iii. Benefit from having cache because it is faster than RAM, and closer to the CPU
        iv. However, cache is smaller than RAM, and costs more per bit than RAM
    b. Terminology
        i. Memory *block* – unit of main memory stored in a cache line
        ii. Cache *line* – basic unit of a cache
            1. Contains a block of memory
            2. Also contains a tag, and control bits to determine when line should be replaced
        iii. Cache *tag* – number stored with a line
            1. Along with the line's position in the cache, determines the address of the block from main memory
        iv. Cache *hit* – data that the CPU asks for is in the cache
        v. Cache *miss* – data that the CPU asks for is *not* in the cache
        vi. *Eviction* – removing memory from the cache and replacing with new memory
        vii. *Dirty* – line contains updated data that differs from the corresponding main memory
            1. Before evicting the line from cache, must copy back to main memory
            2. Otherwise, would lose changes to that data

2. Cache design
    a. Cache design
        i. Multiple-level caches
            1. See these in many modern CPUs
            2. L1 cache is the smallest and closest to the CPU
            3. L2 cache is larger than the L1, but is further away from the CPU
                a. Takes more time to access the L2 than the L1
            4. Same applies for L3
                a. Even larger than L2, but even slower
        ii. Unified or split
            1. Split - have separate caches for data or instructions
            2. Unified – combine the two
    b. How to address the cache
        i. Cache addresses can be *physical*, using actual memory addresses
            1. Slower, but avoids aliasing problem below
        ii. Can also be *logical*, using the same virtual addresses that the program uses
            1. Memory management unit (MMU) converts virtual to physical addresses
            2. Faster, doesn't need to wait for MMU
            3. Encounters the *aliasing* problem
                a. Same virtual address can be used in multiple user processes
                b. Cache would say there's a hit
                c. However, that data doesn't correspond to process currently running
                d. Need to flush cache every time we switch processes
                e. Incur a performance penalty every time we flush cache
    c. Cache size
        i. Larger cache means slower cache
        ii. There is no optimum cache size
            1. Exact specifications will always depend on machine being run