**Q1. cache policies**

Write-back caches are also usually write-allocate because write-back caches don't write to memory until the eviction of a dirty block, meaning that multiple writes are first gathered in the cache. Write-allocate ensures that data being written on a miss is first loaded into the cache, so that writing to a block will happen directly in the cache and not immediately update main memory. If write-back caches were to use the write-non-allocate method, write misses wouldn't go through the cache and would instead write directly to memory, negating the benefits of consolidating writes and reducing memory traffic. Writing to lower level caches and main memory takes time and can be redundant, so the ability to reduce unnecessary writes is crucial.


**Q2. cache performance**

L1 hit rate: 0.8, L1 hit time: 2ns
L1 miss rate: 1 - 0.8 = 0.2
L1 miss time: 2ns (identify a miss) + 8ns (L2 access) = 10 ns
Average memory latency
= (0.8 * 2ns) + (0.2 * 10ns)
= 1.6 ns + 2.0ns
= 3.6 ns

**Q3. virtual memory layout**

64-bit machine, 32GB physical memory, pages are 256KB

a) # virtual pages = # virtual addresses / page size
   = (2^64) / 256 * 2^10
   = (2^64) / (2^18)
   = 2^46 virtual pages per process
b) To uniquely identify 2^46 pages → 46 bits needed
c) # physical pages = physical memory / page size
   = 32 GB / 256 KB
   = 32*(2^30) / 256*(2^10)

= (2^5)*(2^30) / (2^8)*(2^10)
= (2^35) / (2^18)
= 2^17 physical pages

d) To uniquely identify 2^17 pages → 17 bits needed

e) single PPN (16 bits) + valid bit (1 bit) = 17 bits
17 rounded to nearest power of 2 number of bytes → 32 bits, or 4 bytes

f) Page table size = # PTE * size of PTE
= 2^46 virtual pages/process * 4 bytes per PTE
= 2^46 * 2^2
= 2^48 bytes

g) 2^48 bytes of page table space for a single process is simply too much. Flat page tables aren't used for a 64-bit system like this because the amount of physical memory needed for just the page table isn't feasible. To address this issue, multi-level page tables are used. This hierarchical approach segments the page table into multiple levels, where each level indexes a portion of the table, allocating memory only for the parts of the table that are actually used. In contrast, a flat page table would allocate memory for every potential virtual address upfront, even if many of them are unused. This makes multi-level page tables significantly more memory-efficient, especially for large virtual address spaces.

h) Not necessarily. A TLB miss occurs when a virtual-to-physical mapping isn't found in the TLB, which stores recently accessed translations. A TLB is essentially a cache for the page table. A miss does not always lead to a page fault—if the mapping exists in the page table, it can be fetched and loaded into the TLB. A page fault only occurs if the mapping is not found in the page table, meaning the page is not loaded in physical memory, and the operating system must handle the page fault by retrieving the page.