Cache Flush (Write-back)

* Purpose: Ensure that any modified data (dirty cache lines) in the cache are written back to main memory so that memory has the most recent data.
* Action: The cache controller takes all modified cache lines and writes them back to main memory.
* Effect:
  + After a flush, memory is guaranteed to have the most recent data. The cache may or may not retain the data (depends on the architecture).
* Use Case:
  + When a peripheral device (like DMA or GPU) needs to access the latest data from memory. Before context switching in OS (to avoid stale data).

Cache Invalidation

* Purpose: Invalidate (mark as invalid) a cache line so that the next access to that data forces a read from main memory.
* Action: The cache controller marks the cache line as invalid without writing it back to memory.
* Effect:
  + If the CPU accesses the same memory location again, it will trigger a cache miss and re-fetch the data from memory. Prevents stale or outdated data from being used.
* Use Case:
  + After a DMA (Direct Memory Access) operation, where an external device writes to memory, you don't want the CPU to use stale data. Ensuring memory coherence in multiprocessor systems.

Cache Flush + Invalidation (Clean + Invalidate)

* Purpose:
  + First, write back any modified cache lines to memory (flush). Then, mark the cache line as invalid (invalidate).
* Action:
  + Ensures memory has the most recent data. Guarantees no stale data in cache.
* Use Case:
  + Before handing over a memory region to another process/device. Before restarting a hardware context (like GPU buffers).

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Write Data to Memory? | Mark Cache as Invalid? | Use Case |
| Flush (Clean) | ✅ Yes (if modified) | ❌ No | Before DMA read, process switch, ensuring memory has latest data |
| Invalidate | ❌ No | ✅ Yes | After DMA write, prevent stale cache data |
| Flush + Invalidate | ✅ Yes (if modified) | ✅ Yes | Before hardware handover, ensuring memory consistency |

Interview Pro Tip

* Always flush before invalidation if your data is modified and needs to be shared with another device (like DMA).
* Invalidation without a flush may cause data loss if cache lines were dirty (modified but not written back).
* ARM architectures often combine flush + invalidate to ensure cache coherence.