The L1, L2, and L3 caches are crucial components in modern computing architectures, playing a pivotal role in bridging the speed gap between the ultra-fast CPU and the relatively slower main memory (RAM). Let's have a overview on these cache levels  
  
1. 𝑳𝒐𝒄𝒂𝒕𝒊𝒐𝒏 & 𝑷𝒓𝒐𝒙𝒊𝒎𝒊𝒕𝒚:  
  
𝑳𝟏: Directly integrated into the processor core, ensuring minimal access latency.  
It's the primary cache used by the CPU to store instructions and data for immediate processing.  
𝑳𝟐: L2 cache is either located on the CPU chip or situated very close to it on the same die.  
𝑳𝟑: Located on the CPU die but may serve multiple cores, making it a shared resource.   
  
2. 𝑺𝒊𝒛𝒆 & 𝑺𝒑𝒆𝒆𝒅:  
  
𝑳𝟏: ~16KB to 128KB per core. Latency is ~ 0.5 to 1.5 ns. Offers the fastest access time due to its proximity to the CPU core.  
𝑳𝟐: ~256KB to 512KB per core, but can vary. Latency is ~ 5 to 14 ns.  
𝑳𝟑: Ranges widely from 2MB to 50MB or more, shared across all cores. Latency, ~2 to 50 ns, depending on the architecture and distance from the core.  
  
3. 𝑺𝒕𝒓𝒖𝒄𝒕𝒖𝒓𝒆 & 𝑶𝒓𝒈𝒂𝒏𝒊𝒛𝒂𝒕𝒊𝒐𝒏:  
  
𝑳𝟏: Often split into two:  
  
**I-Cache (Instruction Cache)**: Dedicated to holding the upcoming instructions for the CPU. Often uses a direct-mapped or 2-way set associative structure.  
  
**D-Cache (Data Cache):** Contains data for instructions. Typically uses a 2-way or 4-way set associative structure. Cache line/block size is typically 32 or 64 bytes.  
  
𝑳𝟐: Can be unified (holding both data and instructions) or split like L1.  
Typically uses a more highly associative structure than L1, like 8-way or 16-way set associative.  
Similar to L1, typically 32 or 64 bytes for cache line size.  
  
𝑳𝟑: Typically unified. More highly associative than L2, with designs like 16-way or 32-way set associative structures being common. Generally has 64 bytes for cache line size.  
  
4. 𝑹𝒆𝒑𝒍𝒂𝒄𝒆𝒎𝒆𝒏𝒕 𝑷𝒐𝒍𝒊𝒄𝒊𝒆𝒔:  
  
𝑳𝟏: Common algorithms like Least Recently Used (LRU) are employed to decide which entries to evict when new data is brought in. May use write-through or write-back strategies for handling writes.  
  
𝑳𝟐: Common algorithms like Least Recently Used (LRU) are employed to decide which entries to evict when new data is brought in. May use write-through or write-back strategies for handling writes.  
𝑳𝟑: L3 often incorporates advanced prefetching algorithms to anticipate data needs.  
  
Some architectures use a portion of L3 as a "victim cache" for data evicted from L1 or L2, providing a second chance before data is fetched from the slower main memory. Various write policies are employed, similar to L1 and L2.

