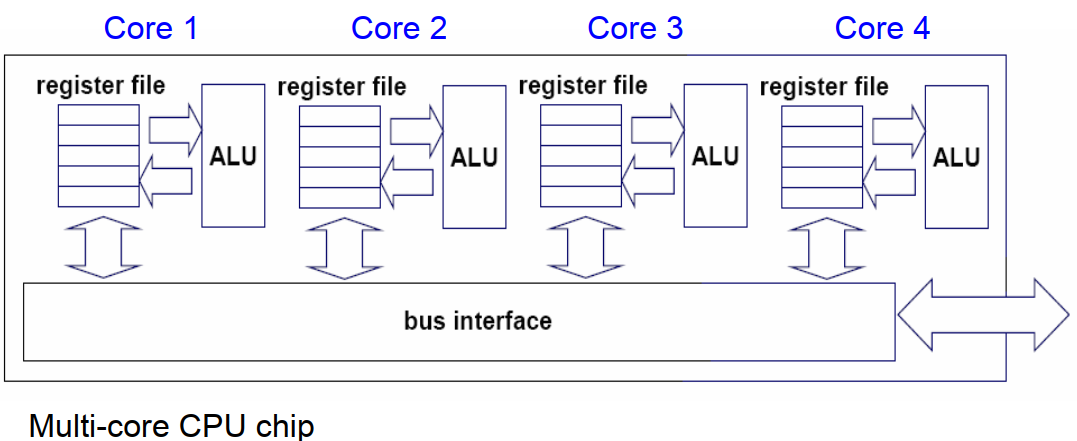
**Multi-core processors**

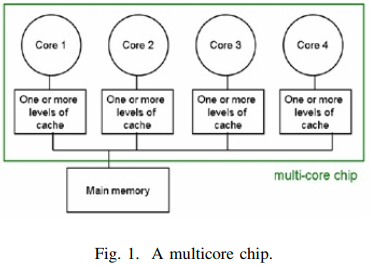
A single processor with multiple cores on a single chip is called a multi-core processor. The cores are functional groups made up of caches and compute units. This single chip's numerous cores work together to mimic the performance of a single faster CPU. Although a multi-core processor's individual cores may not always operate as quickly as the best single-core processors, they do enhance overall performance by handling more jobs concurrently. By knowing how single core and multi-core processors run applications, the speed improvement can be witnessed. When using a single core processor to run many applications, each program would be given its own time slice, with various time slices being given to the other programs. All of the other processes begin to lag behind if one of them takes longer than the others to finish. However, with multi-core processors, if you have several jobs that may be carried out concurrently in parallel, each of them will be carried out by a distinct core in parallel, enhancing performance. Instead of being clocked at a higher frequency, the chip's numerous cores' capacity to run programs simultaneously is what ultimately contributes to overall performance, making them more power-efficient and energy-efficient cores. Generally speaking, multi-core processors are partitioned so that the unused cores can be turned on or off as needed by the program, resulting in overall power dissipation savings.

Depending on the needs of the application, there are numerous approaches to deploy multi-core CPUs. It could be implemented as a collection of homogeneous cores, a collection of heterogeneous cores, or a collection of both. In a homogeneous core design, all of the CPU's cores are the same and use a divide-and-conquer strategy to boost total processor performance by splitting up computationally heavy tasks into less intensive ones and running them concurrently. Another important advantage of employing a homogeneous multi-core processor is that it is easier to meet time-to-market requirements due to its reduced design complexity, reusability, and verification work. Contrarily, heterogeneous cores address the problem of running a range of programs on a computer by utilizing specialized application-specific processor cores. A DSP core that handles multimedia applications requiring intense mathematical computations, a complicated core that handles computationally demanding applications, and a remedial core that handles less computationally difficult applications are a few examples.



In a single physical package, a multi-core CPU implements multiprocessing. Cores of a multi-core device may be coupled firmly or loosely by the designers. For instance, cores may or may not implement message forwarding or shared memory inter-core communication techniques, and they may or may not share caches. Network topologies including bus, ring, two-dimensional mesh, and crossbar are frequently used to link cores. Only identical cores are present in homogeneous multi-core systems; in contrast, non-identical cores (such as large. AMD Accelerated Processing Units have cores that don't even share the same instruction set, while LITTLE have heterogeneous cores that share the same instruction set. Multi-core systems' cores can implement architectures like VLIW, superscalar, vector, or multithreading, just like single-processor systems can. In various application fields, including general-purpose, embedded, network, digital signal processing (DSP), and graphics, multi-core processors are frequently employed (GPU)

**Internal Structure:** Multicore processors are systems with a central processing unit separated into several logical cores, each of which may have one or more private caches, as seen in figure 1.



In the diagram, there are four cores, each of which have one or more caches and is linked to the system's memory through an interconnection network.

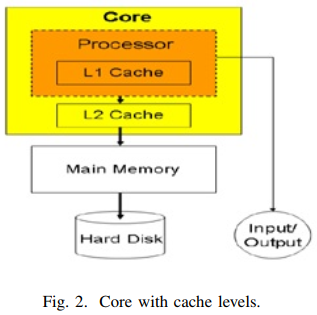


Figure 2 depicts an inside view of a single core. Figure 2 depicts a single-core CPU with a private L1 cache and a shared L2 cache [1, 2]. The distance from the main memory or the number of cycles required to access the main memory determines how many cache layers are needed. Homogeneous and heterogeneous multicore processors are the two different types of multicore CPUs. All of the cores of homogeneous multicore processors are identical, but heterogeneous multicore processors only have non-identical cores.

**Architecture of any multicore processor**

The multicore processor architecture may be roughly categorized into the following categories: application class, power/performance, processing elements, memory system, and accelerators/integrated peripherals.

1. **Application Class:** This multicore processor's architecture is primarily focused on meeting the needs of a certain application domain. This has a number of beneficial impacts, but frequently the multicore for a specific application domain cannot be used or has negative implications when used in other domains. Applications can be classified as either data processing- or control processing-dominated during their execution phase. Data-dominated procedures involve little to no data reuse and are carried out on a collection of data. This fact enables parallel processing with excellent throughput and performance for huge data sets. Examples of applications include wireless base band processing, audio processing, and picture processing. The application deals with a high level of conditional branching, parallelism, and data reuse in the control processing-dominated class. For instance, query processing, network processing, and data compression and decompression.
2. **Power/Performance:** Certain applications demand high power and performance standards. For instance, on Playstations, the game offers a real-time setting. Only applications that leverage multicore architecture with first-class performance design limitations may deliver this feeling. However, as many games require a long battery life for use on mobile devices, power is also a concern for these applications. Therefore, for such applications, the processor's architecture should be such that less power is consumed while still producing excellent performance.
3. **Processing Element:** The type of instruction set architecture, such as Reduced Instruction Set Computer (RISC) or Complex Instruction Set Computer, determines the multicore architecture (CISC). The architecture of a multicore processor is also defined by the processing element used in the core. There are two different sorts of cores based on the processing element: in-order cores and out-order cores. The device size and power consumption of in-order cores are modest, and they can operate with large applications that have more sensitive serial sections and higher levels of parallelism with ease. Outorder cores require additional die space and are not appropriate for systems that use less power.
4. **Memory System:** The intrachip interface, consistency model, cache coherence support, and caches and their levels are all part of the memory system architecture. These decide how the cores will communicate, giving the system a high degree of parallelism and programmability. In essence, the consistency model specifies the order in which the instructions should be carried out. While weak consistency models are less complex and simpler to construct memory systems, strong consistency models include stringent ordering limitations. The cache setup essentially controls how many, how many tiers, and how much of a cache the system needs. The application determines how much cache is needed from an architectural perspective. The size of the cache will increase as data reuse increases. Better performance is a result of larger caches, but it also affects the die area and power budget. How close each processing element is to the main memory determines how many cache levels are necessary. Cache coherence and general communication between CPU components are handled through the intrachip connection. Bus, ring, crossbar, and network on chip comprise the connection for intercore communication. Cache coherence is a requirement for the programming paradigms that the architecture supports. The consistency of data that is visible to all other processor cores is defined by cache coherence.

**Issues in designing Multicore processor**

1. **Cache related issue**
   1. **Amount of cache**: The amount of cache space used by a multicore CPU depends on the application. Large caches are desirable for applications where data reuse is high. Greater performance will result from larger cache sizes, but the price will be higher. Larger cache sizes also result in faster access times.
   2. **Number of cache level**: Choosing how many cache levels a multicore cache can have is another caching issue. It is not necessary for each cache level in the core's cache to be equal. In essence, the distance from the main memory or the number of cycles required to access the main memory determines how many cache levels are needed. The cache levels will be higher and access times will be quicker as the number of cycles increases.
2. **Selection of core used:** a program divides up its duties among the processor's cores in order for the tasks to compute effectively. Regarding this, two concerns must be resolved: choosing the type of core that is compatible with the duties that are allocated to them and the number of cores required for a certain application.
3. **Consistency among cores:** The copy of a piece of data in each cache might not be the same because each core has its own private cache. Cache coherence, or consistency among cache, ensures that the other CPU cores see the same image of data stored in memory when performing computations. Both broadcast coherence and directory-based coherence can be used to implement cache coherence [3]. Only one processor may conduct an operation in broadcast coherence.
4. **Speed vs.** **Power:** Applications today need great performance, which necessitates quick computing speeds. If additional cores are employed or the cache capacity is larger, a multicore processor's speed can be boosted. The system will need more static and dynamic power if the number of processing elements is raised. If all the cores operate at their full potential, overheating may also be an issue. Therefore, methods or plans should be used to obtain excellent performance at a lower rate of power consumption.