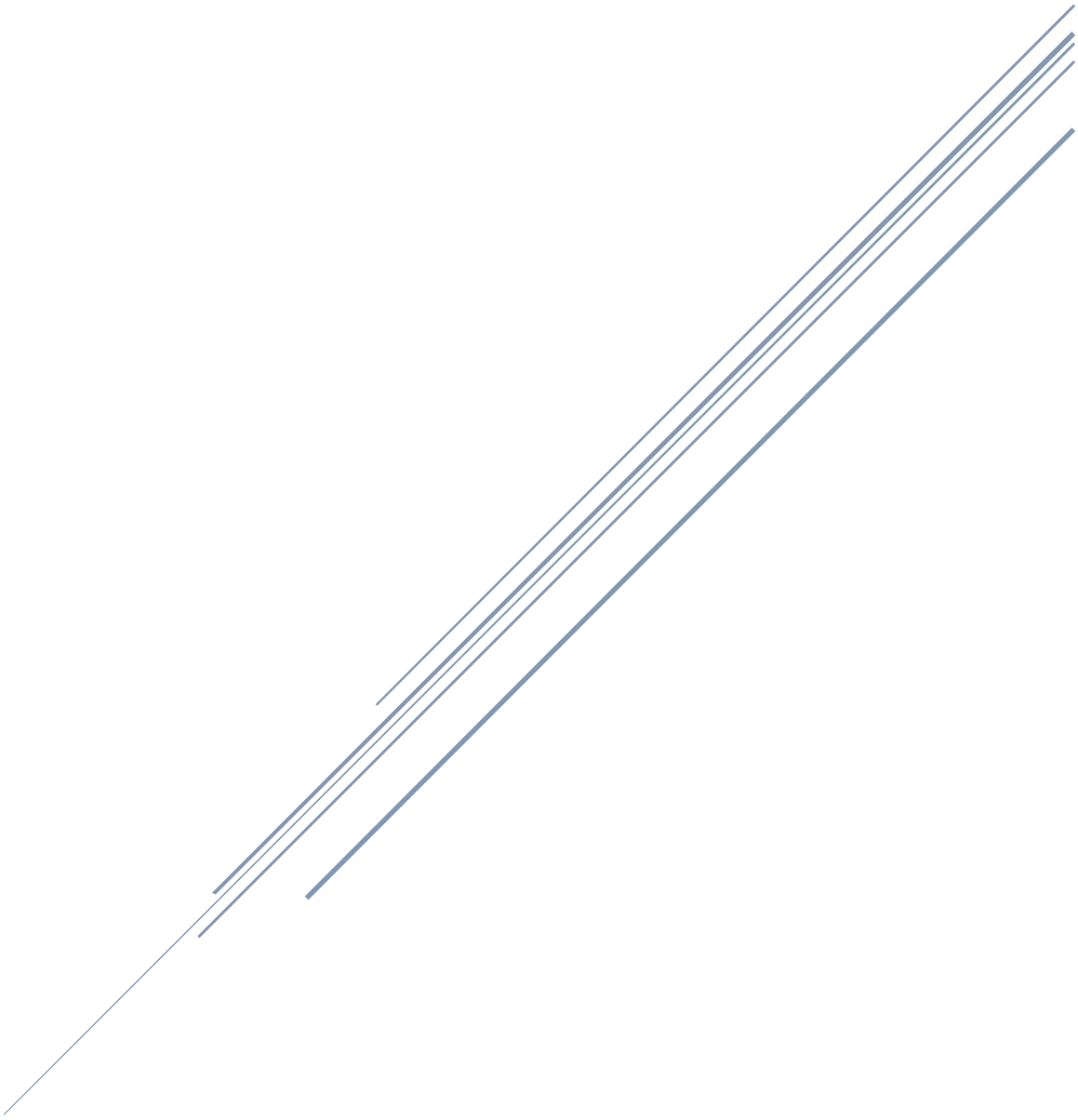
MULTICORE COMPUTERS



Abdallah Aljayeh	201930047
Anas Alseid	202020067
Mohammed Haboush	201820148
Obada Moqbel	202010367

Multicore computers

Multicore computers are computers that have more than one central processing unit (CPU), on their microprocessors that contain multiple independent processing units known as cores. These additional cores can be used to perform multiple tasks concurrently. They work in such a way that the computer system appears to have several processors, improving the overall performance of the computer.

Hardware performance

Multicore computers can generally perform tasks faster than single-core computers due to their ability to divide the workload among multiple cores. By placing two or more processor cores on the same device, it can use shared components such as common internal buses and processor caches more efficiently.

This can lead to significant performance gains for tasks that can be parallelized, such as video rendering and scientific simulations. However, the Performance boost is limited. The more processor cores share a package, the more sharing must take place across common processor interfaces and resources. This results in diminishing returns to performance as cores are added. For most situations, the performance benefit of having multiple cores far outweighs the performance lost to such sharing.

Software performance

The benefits of multicore computers depend on the software's ability to take advantage of the multiple cores. If the software is not designed to use multiple cores, it will only be able to use a single core and will not see any performance improvements. On the other hand, if the software is designed to use multiple cores, it can potentially see significant performance improvements on a multicore computer.

Overall, the performance of multicore computers depends on both the hardware and the software being used. To get the most out of a multicore computer, it is important to use software that is optimized for multicore systems.

The multicore organization (IBM)

Multicore organization refers to how multiple CPU cores are arranged and connected within a computer. There are several different ways that multicore processors can be organized, and the specific organization can have a significant impact on the performance of the computer.

Symmetric Multiprocessing (SMP)

One common multicore organization is known as symmetric multiprocessing (SMP). In an SMP system, all the cores are connected to a single shared memory and can access the same data. This allows any of the cores to work on any task and can lead to good performance for tasks that can be parallelized. An example of a multicore processor with an SMP organization is the IBM power8, which is a 64-bit processor with up to 12 cores. Each of the cores in power8 is fully independent and can access shared memory, allowing them to work on tasks concurrently.

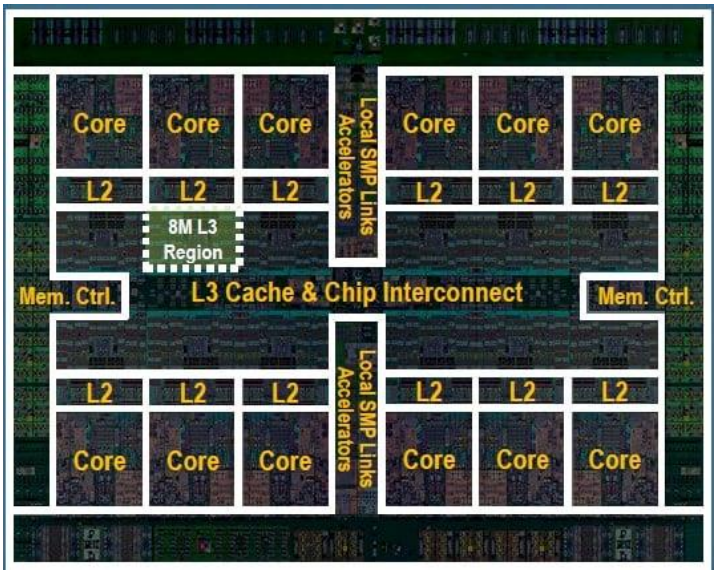


Figure 1 Block diagram of the Power8 chip

Non-Uniform Memory Access (NUMA)

Another common multicore organization is known as non-uniform memory access (NUMA). In a NUMA system, the cores are divided into groups, and each group has its own local memory. NUMA moves the data faster while lowering the latency in the system and reducing data replication. This can lead to improved performance for tasks that frequently access the local memory but can also result in slower performance for tasks that need to access memory in another group.

An example of a multicore processor with a NUMA organization is the IBM power9, which is a 64-bit processor with up to 24 cores. The power9 is divided into two groups of 12 cores, each with its own local memory. This allows the cores to work on tasks concurrently, with good performance for tasks that access the local memory and potentially slower performance for tasks that need to access memory in the other group.

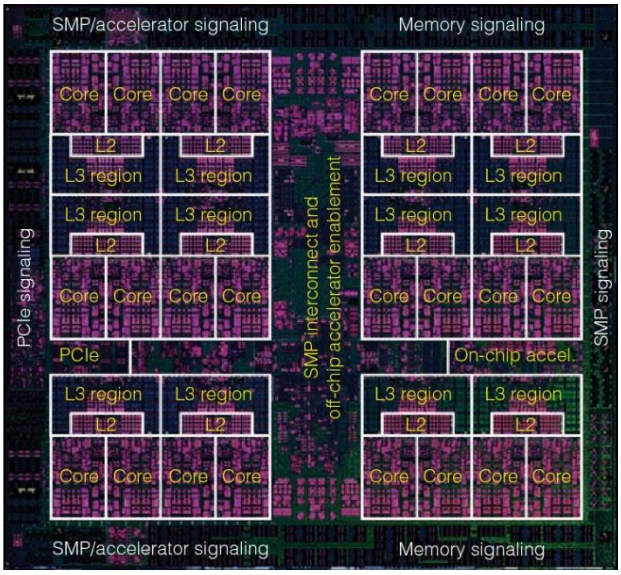


Figure 2 Block diagram of the Power9 chip

Overall, the specific multicore organization of a processor can have a significant impact on its performance. SMP systems are generally good for tasks that can be parallelized, while NUMA systems can be better for tasks that access local memory frequently.