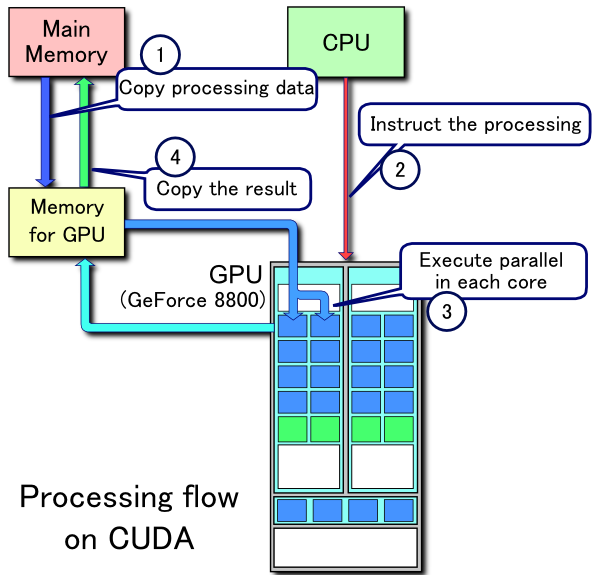
**CUDA** is a parallel computing platform and application programming interface (API) model created by Nvidia. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing – an approach termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

The CUDA platform is designed to work with programming languages such as C, C++ and Fortran. This accessibility makes it easier for specialists in parallel programming to use GPU resources, in contrast to prior APIs like Direct3D and OpenCL, which required advanced skills in graphics programming. Also, CUDA supports programming frameworks such as OpenACC and OpenCL. When it was first introduced by Nvidia, the name CUDA was an acronym for **Compute Unified Device Architecture**, but Nvidia subsequently dropped the use of the acronym.



Sorting is a very important task in computer science and becomes a critical operation for programs making heavy use of sorting algorithms. General-purpose computing has been successfully used on Graphics Processing Units GPUs to parallelize some sorting algorithms. Two GPU-based implementations of the quicksort were presented in literature: the GPU-quicksort, a compute-unified device architecture CUDA iterative implementation, and the CUDA dynamic parallel CDP quicksort, a recursive implementation provided by NVIDIA Corporation. We propose CUDA-quicksort an iterative GPU-based implementation of the sorting algorithm. CUDA-quicksort has been designed starting from GPU-quicksort. Unlike GPU-quicksort, it uses atomic primitives to perform inter-block communications while ensuring an optimized access to the GPU memory. Experiments performed on six sorting benchmark distributions show that CUDA-quicksort is up to four times faster than GPU-quicksort and up to three times faster than CDP-quicksort. An in-depth analysis of the performance between CUDA-quicksort and GPU-quicksort shows that the main improvement is related to the optimized GPU memory access rather than to the use of atomic primitives. Moreover, in order to assess the advantages of using the CUDA dynamic parallelism, we implemented a recursive version of the CUDA-quicksort. Experimental results show that CUDA-quicksort is faster than the CDP-quicksort provided by NVIDIA, with better performance achieved using the iterative implementation.

In computing, an **odd–even sort** or **odd–even transposition sort** (also known as **brick sort** is a relatively simple sorting algorithm, developed originally for use on parallel processors with local interconnections. It is a comparison sort related to bubble sort, with which it shares many characteristics. It functions by comparing all odd/even indexed pairs of adjacent elements in the list and, if a pair is in the wrong order (the first is larger than the second) the elements are switched. The next step repeats this for even/odd indexed pairs (of adjacent elements). Then it alternates between odd/even and even/odd steps until the list is sorted.

## Sorting on processor arrays

On parallel processors, with one value per processor and only local left–right neighbor connections, the processors all concurrently do a compare–exchange operation with their neighbors, alternating between odd–even and even–odd pairings. This algorithm was originally presented, and shown to be efficient on such processors, by Habermann in 1972.

The algorithm extends efficiently to the case of multiple items per processor. In the Baudet–Stevenson odd–even merge-splitting algorithm, each processor sorts its own sublist at each step, using any efficient sort algorithm, and then performs a merge splitting, or transposition–merge, operation with its neighbor, with neighbor pairing alternating between odd–even and even–odd on each step.