1. Task: a task is a program-like set of instructions that is executed by a processor. In the case of parallel programming, multiple tasks are run on multiple processors simultaneously.

Pipelining: it consists in dividing a task into steps that will be executed by different processor units, with inputs streaming through those steps.

Shared Memory: when referring to hardware, shared memory refers to when a computer’s processors all have access to a common physical memory. However, from a programming point of view, it refers to a model where parallel tasks all can access the same logical memory locations regardless of where the physical memory is located.

Communications: it’s the data exchanges of parallel tasks that can be accomplished through different ways like through a shared memory bus or a network.

Synchronization: the coordination of parallel tasks in real time. It is usually implemented by choosing a point within the application that a task cannot go past another task get to the same point or a logically equivalent point.

1. Parallel computers can be classified as SIMD (Single Instruction, Multiple Data), MISD (Multiple Instruction, Single Data) and MIMD (Multiple Instruction, Multiple data), according to Flynn’s Taxonomy.

In SIMD parallel computers, the same instruction executes by all processing units at any given clock cycle, but each processing unit can use a different data element. It is best suited for problems that follow a path of regularity because of its characteristics.

In the case of MISD, it is also a type of parallel computer, but the processing units operate via separate instruction streams. It is single data because a single stream of data is fed into multiple processing units. This type of computer has very few applications and it is not in use.

Lastly, in MIMD computers, the processing units execute different instruction streams with different data. Execution can be both synchronous or asynchronous, deterministic or not deterministic. Supercomputers fall in this classification.

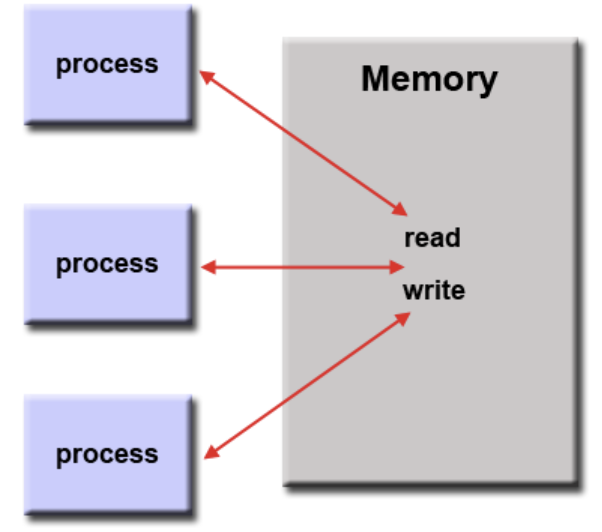
1. The most commonly used parallel programming models are Shared Memory, Threads, Distributed Memory/Message Passing, Data Parallel, Hybrid, Single Program Multiple Data (SPMD) and Multiple Program Multiple Data (MPMD).
2. The types of parallel computer memory architectures are: Uniform Memory Access (UMA) and Non-Uniform Memory Access (NUMA).

UMA architectures have identical processors, each with equal access and access time to memory. It is sometimes called Cache Coherent UMA (CC-UMA), because if one processor updates a location in the shared memory, the other processors are aware of the update. It is commonly represented by Symmetric Multiprocessor (SMP) machines.

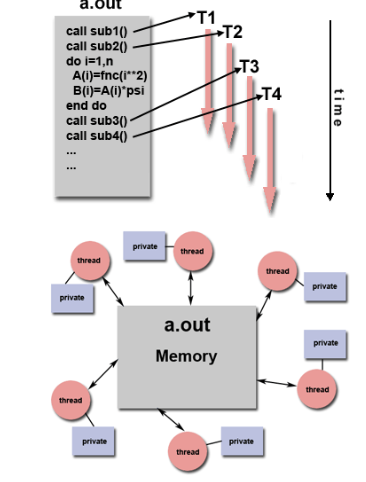
NUMA architectures are usually made by two or more SMPs where one can access directly access the memory of another SMP, and processors do not have equal access time to all memories. They can also be cache-coherent (CC-NUMA).

OpenMP uses UMA Computer architectures because the architecture of those computers are more suitable to run parallel processes. In UMA, single, multiple and crossbar busses are used and NUMA uses hierarchical and tree type of busses and network connection. Also, the memory accessing time for each processor is the same in UMA while, in NUMA, the memory accessing time changes as the distance of memory from the processor changes.

1. In the Shared Memory Model, all processes share a common address space that they can read and write asynchronously. It’s a very simple parallel programming model but it is harder to control and manage data locality.



In the Threads Model the process is divided into subroutines called threads. When the main program runs, it collects the system and user resources and then creates threads to execute tasks that can run parallely. The threads share the memory with the main program but each thread also has local data.



1. Parallel Programming is a form of programming that uses multiple processors or processor’s cores to divide processes into smaller subroutines that execute simultaneously, offering the same result but in less time and more efficiently.
2. A system on chip (SoC) integrates all the components of a CPU into a single silicon chip and can function without the addition of any other chips. All of the Raspberry Pi model use SoC.
3. The advantages of using a system on chip are that it contains more functionality for its size when compared to a CPU, which allows to build highly functional yet small devices like smartphones or tablets. SoC also uses less power than a CPU to run and because it has less added components, it is also cheaper to build a computer using SoC.