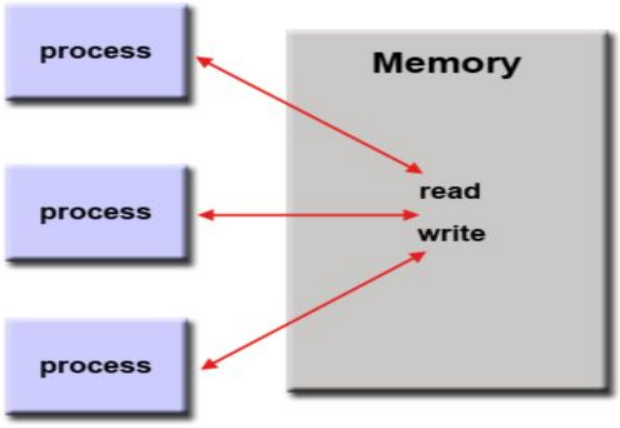
List and briefly describe the types of Parallel Computer Memory Architectures. What type is used by OpenMP and why?

Basically, the main memory in a parallel computer is either Shared Memory or Distributed Memory. The Shared Memory means that the memory is shared between all processing elements in a single address space while each processing element has its own local address space in distributed memory. In fact, the logically distributed memory is often implied as physically distributed. Distributed shared memory and memory virtualization combine the two approaches, where the processing element has its own local memory and access to the memory on non-local processors. Accesses to local memory are typically faster than accesses to non-local memory.

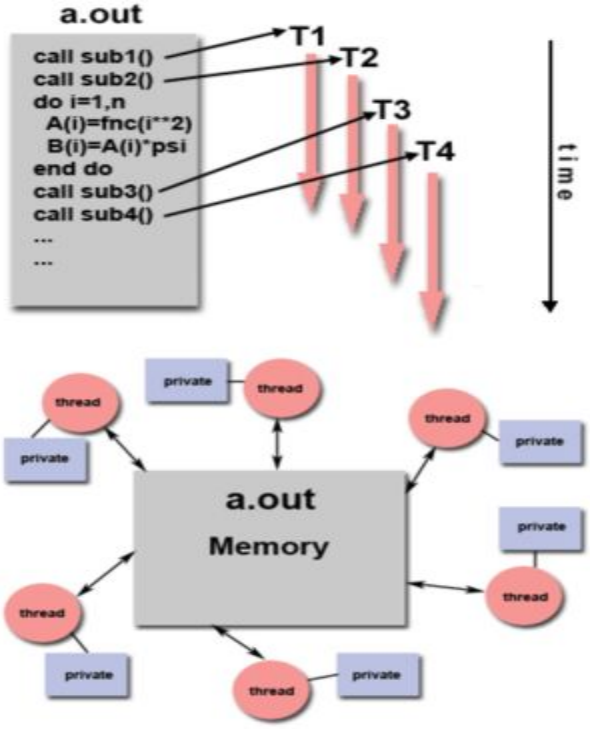
OpenMP uses shared memory, or more specifically, OpenMP uses Threads model of Shared Memory. It is designed for multi-processor/core shared memory machines.

Compare Shared Memory Model with Threads Model? (In your own words and show pictures)

For shared memory model, processes/tasks share a common address space, which they read and write to asynchronously. And to control access to the shared memory, resolve contentions and to prevent race conditions and deadlocks, various mechanisms such as locks / semaphores are used. The advantage of shared memory model from the programmer's point of view is that the notion of data "ownership" is lacking, so there is no need to specify explicitly the communication of data between tasks. All processes have equal access to shared memory which will simplify program development. However, in terms of performance, it becomes more difficult to understand and manage data locality. Although this model keeps data local to the process that works on it conserves memory accesses, cache refreshes and bus traffic that occurs when multiple processes use the same data, it may be beyond the control of the average user. Typical implementations is that on distributed memory machines, memory can be accessed globally through specialized hardware and software. The following figure is an interpretation of shared memory model.



Thread model is a type of shared memory programming. A thread's work may best be described as a subroutine within the main program with any thread executing any subroutine at the same time as other threads. In this model, a single "heavy weight" process can have multiple "light weight" concurrent execution paths. For example, the main program a.out is scheduled to run by the native operating system, and a.out loads and acquires all of the necessary system and user resources to run (This is the "heavy weight" process), and a.out performs some serial work, then creating a number of tasks (threads) that can be scheduled and run by the operating system concurrently (This is the "light weight" process). In thread model, each thread shares the same resource which saves the overhead associated with replicating the source, so each thread can benefit from a global memory view. Threads communicate with each other through global memory (updating address locations), but this requires synchronization constructs to ensure that more than one thread is not updating the same global address at any time. Also, threads can come and go, but the main program remains present to provide the necessary shared resources until the application has completed. Two important implications are POSIX Threads and OpenMP due to different standards. The following figure is an interpretation of threads model.



What is Parallel Programming? (In your own words)

Parallel computing is the simultaneous use of multiple compute resources to solve a computational problem under the conditions that 1) the problem is broken into discrete parts that can be solved concurrently; 2) and each part is further broken down to a series of instructions, or

the instructions from each part can be executed simultaneously on different processors; 3) employing an overall control/coordination mechanism.
For the problem to be solved, it should be able to: 1) be broken apart into discrete pieces of work that can be solved simultaneously; 2) multiple program which can be executed instructions at any moment in time; 3) be solved in less time with multiple compute resources than with a single compute resource.
A typical compute resource would be a single computer with multiple processors/cores, and an arbitrary number of such computers connected by a network can also work as resources.

What is system on chip (SoC)? Does Raspberry PI use system on SoC?
The full name of SoC is system-on-a-chip. SoC integrates almost all of these components into a single silicon chip, and the basic components contain a CPU, a GPU (a graphics processor), memory, USB controller, power management circuits, and wireless radios (WiFi, 3G, 4G LTE, and so on). Compared to SoC, a single CPU cannot function without dozens of other chips. So it's possible to build a computer with SoC.
Raspberry PI uses SoC which contains ARM1176JZF-S 700MHz CPU with 512MB SDRAM.

Explain what the advantages are of having a System on a Chip rather than separate CPU, GPU and RAM components.
The advantages of using a SoC are smaller sizes and using less power. Considering that SoC has more functionality, it's only a little bit larger than a single CPU, and it uses less power due to high level of integration and much less wiring. Although lacking flexibility, all these advantages still make SoC a popular choice for portable devices, such as smartphones and tablets.