1) Define the following:

1. Task: A task is typically a program or program like set of instructions that is executed by a processor. A parallel program consists of multiple tasks running on multiple processors
2. Shared Memory: It describes a computer architecture where all processors have direct access to common physical memory. In a programming sense, it describes a model where parallel task all have the same "picture" of memory and can directly address and access the same logical memory locations regardless of where the physical memory actually exists.
3. Pipelining: Breaking a task into steps performed by different processor units, with inputs streaming through, much like an assembly line; a type of parallel computing.
4. Communications: Parallel tasks typically need to exchange data. There are several ways this can be accomplished, such as through a shared memory bus or over a network, however the actual event of data exchange is commonly referred to as communications regardless of the method employed.
5. Synchronization: The coordination of parallel tasks in real time, very often associated with communications. It is implemented by setting a ‘synchronization point’ within an application where a task may not proceed further until another task(s) reaches the same or logically equivalent point. This can cause the time of execution to increase.

2) Flynn’s taxonomy:

a) Single Instruction, Single Data (SISD):

• A serial (non -parallel) computer

• Single Instruction: Only one instruction stream is being acted on by the CPU during any one clock cycle

• Single Data: Only one data stream is being used as input during any one clock cycle.

b) Single Instruction, Multiple Data (SIMD):

• A type of parallel computer

• Single Instruction: All processing units execute the same instruction at any given clock cycle

• Multiple Data: Each processing unit can operate on a different data element

c) Multiple Instruction, Single Data (MISD):

• A type of parallel computer

• Multiple Instruction: Each processing unit operates on the data independently via separate instruction streams.

• Single Data: A single data stream is fed into multiple processing unit

d) Multiple Instruction, Multiple Data (MIMD):

• A type of parallel computer

• Multiple Instruction: Every processor may be executing a different instruction stream

• Multiple Data: Every processor may be working with a different data stream

3) Parallel Programming Models:

There are several parallel programming models in common use:

O Shared Memory (without threads)

o Threads

o Distributed Memory / Message Passing

o Data Parallel

o Hybrid

o Single Program Multiple Data (SPMD)

o Multiple Program Multiple Data (MPMD)

• Parallel programming models exist as an abstraction above

hardware and memory architectures.

• Although it might not seem apparent, these models are NOT specific to

a particular type of machine or memory architecture. In fact, any of

these models can (theoretically) be implemented on any underlying

hardware.

4) . Types of Parallel Memory Computer Architecture

General Characteristics:

• Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.

• Multiple processors can operate independently but share the same memory resources.

• Changes in a memory location effected by one processor are visible to all other processors.

• Historically, shared memory machines have been classified as UMA and NUMA, based upon memory access times

1. Uniform Memory Access (UMA):

• Most commonly represented today by Symmetric Multiprocessor (SMP) machines

• Identical processors

• Equal access and access times to memory

• Sometimes called CC -UMA -Cache Coherent UMA.

Cache coherent means if one processor updates a

location in shared memory, all the other processors

know about the update. Cache coherency is

accomplished at the hardware level.

1. Non - Uniform Memory Access (NUMA):

• Often made by physically linking two or more SMPs

• One SMP can directly access memory of another SMP

• Not all processors have equal access time to all

memories

• Memory access across link is slower

• If cache coherency is maintained, then may also be

called CC -NUMA - Cache Coherent NUMA

Open MP uses Thread Model

5) Compare Shared Memory Model with Threads Model

SHARED MEMORY MODEL:

\* In this programming model, processes/tasks share a common address space, which they read and write to asynchronously.

\* An advantage of this 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 see and have equal access to shared memory. Program development can often be simplified

\* An important disadvantage in terms of performance is that it becomes more difficult to understand and manage data locality

O Unfortunately, controlling data locality is hard to understand and may be beyond the control of the average user.

THREADS MODEL:

* This programming model is a type of shared memory programming.
* In the threads model of parallel programming, a single "heavy weight" process can have multiple "light weight", concurrent execution paths.
* From a programming perspective, threads implementations commonly comprise:

o A library of subroutines that are called from within parallel source code

o A set of compiler directives imbedded in either serial or parallel source code

6) Parallel Computing:

• In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:

o A problem is broken into discrete parts that can be solved concurrently

o Each part is further broken down to a series of instructions

o Instructions from each part execute simultaneously on different processors

o An overall control/coordination mechanism is employed

7) An SoC, (System on chip) integrates memory, audio and graphics components into a single silicon chip. Along with a CPU, an SoC usually contains a GPU (a graphics processor), memory, USB controller, power management circuits, and wireless radios (Wi-Fi, 3G, 4G LTE, and so on). Whereas a CPU cannot function without dozens of other chips, it’s possible to build complete computers with just a single SoC.

The RAM, CPU, GPU of the Raspberry Pi are all squeezed into a SoC.

8) Advantages of SoC over CPU:

a) Size: An SoC is only a little bit larger than a CPU, and yet it contains a lot more functionality.

b) Due to its very high level of integration and much shorter wiring, an SoC also uses considerably less power, hence it is useful for mobile computing.

c) Cutting down on the number of physical chips means that it’s much cheaper to build a computer using an SoC