**(57p) Foundation (reading material for this section is available at iCollege, Week7, Project: Assigenmnt3 folder, Introduction to Parallel Computing\_2.pdf)**

**(5p)Define the following:**

* **Task**
  + A task is a piece of a larger program that can be executed by a processor. A program is broken into tasks by the OS and they are assigned to the processors to be executed simultaneously.
* **Pipelining**
  + A parallel computing technique where a problem or program is broken into smaller simpler units/tasks and these units are streamed through multiple processors for execution.
* **Shared Memory**
  + There are two levels of shared memory. The first is the Hardware level. Hardware Shared Memory describes the concept of the processors having direct access to the Memory, usually through a Data Bus and Address Bus. The second level is the Programming level. In this level, the concept represents the notion of tasks, the smaller units of the program, having logical access to the Memory when being executed.
* **Communications**
  + When tasks are being executed, they sometimes need to communicate between themselves in order to exchange values or perform complex processes. This communication can take place through a shared memory bus in a computer or, if the parallel computing is through a network, a network bus.
* **Synchronization**
  + The concept involves the notion of balancing the advancement of the process in order to allow tasks to generate the correct results when being executed through a pipeline. A task being executed by a processor may need to access a value from the result of another task. In order to do this, the process may need to be slowed or halted until the desired value is received. This is the synchronization of the process.

**(8p)Classify parallel computers based on Flynn's taxonomy**

* **SISD (Single Instruction/Single Data Stream)**
  + This type of computers are not considered parallel computers. In this type of computer, only one stream of instruction stream and one stream of data stream are being processed at one given time. A single core PC would be one type of a SISD computer.

Machine generated alternative text:
Program 
varA 
varB 
var2 
var3 
processAl phaA() 
Process Numl() 
processAl phaB() 
ProcessNum2() 
ProcessAl phaC() 
ProcessNum3() 
Datastream»»»»»» 
Operating 
System 
Instruction Stream 
Output 
cpu 1 

* Different data and different instructions are processed sequentially in a single processor.

* **SIMD (Single Instruction/Multiple Data Stream)**
  + This type of parallel computers are design to process the same instructions among the multiple processors on different set of data streams. This type of architecture is mostly utilized by graphics/image processing units and cryptography computers, where the same task is being performed on different data.

Machine generated alternative text:
Program 
varA 
varC 
va rl 
var2 
vary 
ProcessAll() 
Operating 
System 
Instruction Stream 
Data Stream 
Instruction Stream 
cpu 1 
CPU 2 
ProcessAll() 
Output 
ProcessAll() 

* Different data is processed by the same instruction on separate processors.

* **MISD (Multiple Instruction/Single Data Stream)**
  + In this type of parallel computers, each processing unit operate on the same data stream independently with a separate instructions. This type of computing is best when processing the same data in different ways.

Machine generated alternative text:
Program 
var 
rocessAlphaA() 
Operating 
ProcessNum1() 
System 
processAlphaB() 
ProcessNum2() 
ProcessAlphaC() 
rpeessNgm3 
Instruction Stream 
var 
Instruction Stream 
CPU 1 
CPU 2 
Output 

* Same data is processed by different instructions on separate processors.

* **MIMD (Multiple Instruction/Multiple Data Stream)**
  + In this type of parallel computers, every processing unit would be executing a different instruction on different data streams. The processing units can be configured to be Deterministic (always producing the same output, similar to a physics simulator) and/or Synchronous (processing units and/or tasks are in constant communication in order to accomplish a certain results).

Machine generated alternative text:
Program 
varA 
varB 
varC 
varl 
var2 
var3 
rocessAlphaA() 
ProcessNum1() 
rocessAlphaB() 
ProcessNum2() 
ProcessAlphaC() 
Num 
Operating 
System 
Instruction Stream 
Data Stream 
Instruction Stream 
cpu 1 
CPU 2 
Output 

* Different data is processed by different instructions on separate processors.

**(7p)What are the Parallel Programming Models?**

* **Shared Memory Model without Threads** also known as **Virtual Shared Memory** is an abstract form of parallel computing implementation where Memory is distributed among different nodes in a network. These nodes share their Memory among each other and act as one physical Memory address space to the user and any running tasks. This is the simplest form of Parallel Programming since there is no communication between tasks and processes have equal access to the Memory.
* **Threads Model** Parallel Programming consist of two parts: a "Heavy Weight" Process and its "Light Weight" sub-processes. The "Heavy Weight" program is loaded in the Operating System where all the preparations, such as system resources, are loaded for the program to run. The program is then broken into sub-tasks, known as Threads. These Threads are distributed among the available processors to run simultaneously. Each thread contains its own local data and also has full access to the Global Variables and all necessary shared resources of the program in the Memory. A synchronization method is applied in order to coordinate all the processes between the Threads.
* **Distributed Memory/Message Passing Model** is a type of parallel programming in which tasks are distributed among different computers in a network. These tasks are setup to communicate with each other through passing of message operations.
* **Data Parallel Model**, also known as **Partitioned Global Address Space Model**, consist of a shared addressable Memory where a global data structure, like an array, is shared among a set of tasks. The tasks work together by processing different parts of the global data structure.
* **Hybrid Model** combine the **Thread Model** and the **Distributed Memory/Message Passing Model**. The nodes process the threads with its local data and process communicate among each other through **Message Passing Model (MPI)** operations.
* **Single Program Multiple Data (SPMD) Model** is considered a "High Level" programming model. The SPMD model that can be implemented on other parallel programming models. The concept involves running different copies of a tasks and each task processing different data. The SPMD programs can be setup to execute only certain parts of a program among different threads.

**(12p)List and briefly describe the types of Parallel Computer Memory Architectures**

* What type is used by OpenMP and why?
* **Shared Memory**
  + **Shared Memory Architecture** depicts the concept of parallel computer processors sharing a common Memory space. The Memory is available to all the processors and any change to the memory is visible to all the processors. There are two categories of representation of the Shared Memory Architecture.
    - **Uniform Memory Access (UMA)** represents models where identical processors have equal access to the Memory and any change to the Memory is known by all the processors. This is accomplished with the utilization of Cache coherency. Symmetric Multiprocessor (SMP) machines are an example of a UMA Architecture.
    - **Non-Uniform Memory Access (NUMA)** represents models where multiple UMA Model computers are connected together in order to share memory access. Every machine in the cluster contains its own processors and memory. These machines are interconnected in order to share their resources (memory) within the cluster. In this model, not all processors have equal access to the memory and memory access is slower.
* I believe OpenMP would fall in the **Uniform Memory Access (UMA)** category since all the processors have full access to the memory and the global variables in it. Changes to the shared memory are visible to all the processors and threads/tasks.

**(10p)Compare Shared Memory Model with Threads Model? (in your own words and show pictures)**

* A Shared Memory Model (without threads) is an abstraction to describe how processes/tasks are to access and store information in a shared memory environment. Shared Memory Model depicts the notion that all processes/tasks are to have a shared memory location. This memory is available to all the processes and information can be accessed and modified asynchronously; every process can function independently. This type of programming model is the simplest to implement and is available for both, stand-alone computers and network computers connected to share memory. For a stand-alone computer, the POSIX standard can be applied in a Unix environment.

Machine generated alternative text:
Progam 
globalArrayzA, B, C, D 
taskA 
varA 1 
add globalArray + varA 
taskB 
varB 2 
Operating 
add globalArray + varB 
System 
taskC 
varC 3 
add globalArray + varC 
taskD 
varD 4 
add globalArray + varD 
Memory 
taskA 
varA 
taskB 
varB 
taskC 
varC 
varD 
taskD 

* A Thread Model is another abstraction of executing parallel programming in a shared memory environment. In this type of model, the Operating System breaks a program into tasks/threads and then runs them concurrently by the processors. Unlike the Shared Memory model, the tasks in the Threads model contain its own local data and also have access to the global data in the shared memory. The advantage of this technique is that each thread/task contains its own resources, such as library and I/O resources. This streamlines the process since resources do not have to be redistributed every time they are called by the tasks. The tasks also run in synchronization and that requires that tasks keep in communication with each other. This communication is done though global variables in the shared memory. The resources of the running program, as well as a copy of the running program, are available to all the tasks in the shared memory until the program ends. The most common implementation of the Thread Model are POSIX Threads and OpenMP.

Machine generated alternative text:
Progam 
globalArrayzA, B, C, D 
taskA 
varA 1 
add globalArray + varA 
taskB 
varB 2 
Operating 
add globalArray + varB 
System 
taskC 
varC 3 
add globalArray + varC 
taskD 
varD 4 
add globalArray + varD 
taskA 
varA 
taskB 
varB 
taskC 
varC 
taskD 
varD 
Memory 
taskA 
globa 'Array 
taskB 
taskC 
varC 
globa 'Amy 

**(5p)What is Parallel Programming? (in your own words)**

* Parallel programming is the technique of the utilizing shared resources, such as memory, and breaking a job/program into smaller units/tasks in order to process them simultaneously in the available processors.

**(5p)What is system on chip (SoC)? Does Raspberry PI use system on SoC?**

* SoC integrates several components into a silicon chip unit. Most SoC integrate the CPU, GPU, Memory, USB, and other components in a single silicon chip.
* System On Chip (SOC) have several advantages over a CPU. Due to its smaller size, SoC consume less power. Their size also allow more components to be mounted in a single silicon chip and adding more functionality to the device. Size also has an impact on manufacturing costs, since having less physical wiring and chip components make the SoC much cheaper than having a device with separate components like CPU, GPU and memory.