Course No	Title of the Course	Course Structure	Pre-Requisite
COCSC18	**High Performance Computing**	**3L-0-2P**	Computer Architecture and Organization

COURSE OUTCOMES

1. To understand the capabilities, limitations and performance of high-performance architectures and their applications in solving challenging problems.
2. Develop the skills to decompose parallelizable problems effectively.
3. Write parallel algorithms and use parallel programming paradigms to implement them.
4. Appreciate the multidisciplinary approach for developing and utilizing high performance systems.
5. Getting the exposure to set up a small cluster and various concepts of computing.

COURSE CONTENTS

UNIT-1

Introduction: Flynn's classification of parallel architectures, Kinds of parallelism- Temporal, data and mixed parallelism , Dependencies and hazards - data, control and resource dependencies, PRAM models.

UNIT-2

Parallel Programming paradigms: Granularity and Communication overheads, Program decomposition techniques, Shared Memory Programming (pthreads), SPMD model, Message Passing Programming (MPI/Open MP), Parallel sorting, even-odd transposition/ parallel multiplication/ Parallel matrix operations on PRAM models.

UNIT-3

High Performance Architectures: Instruction level parallelism-Delays in instruction pipelining, mechanisms to tackle pipeline stalls, superscalar, superpipelined architectures, VLIW processors. Array Processors -SIMD architectures, Vector processing architectures. Multiprocessor architectures shared memory symmetric multiprocessing, clusters and grids. Interconnection networks - characteristics and routing mechanisms. Parallel algorithms on realistic architectures.

UNIT-4

Performance and scalability evaluation: Performance Laws: Amdahl's Law, Guftanson's Law, Sun and Li Law, Performance Benchmarks, Overheads in parallel processing, Hardware software matching

UNIT-5

Memory and cache Consistency: Memory consistency: strict consistency, Lamport's sequential consistency, strong and weak consistency models. Bus based and directory based cache coherence protocols.

Guidelines for practical work:

Shared memory inter-process communication using pthreads – develop applications to demonstrate inter-process communication by thread creation, parameter passing, thread joining using semaphores, mutex and condition

variables.

- Message passing parallel programming – develop applications to demonstrate task partitioning and IPC using Message Passing Interface (MPI) and OpenMPI such as different parallel implementations of matrix multiplication and sorting. Create a simple cluster.

SUGGESTED READINGS

1. Kai Hwang, “Advanced Computer Architecture”, McGrawHill
2. V. Rajaraman and C. Siva Ram Murthy, “Parallel Computers, architecture and programming, PHI
3. Michael J Quinn, “Parallel Programming in C with MPI and OpenMP”, McGrawHill Edu.
4. Peter Pacheco, “An introduction to parallel programming”.