Parallel Program Development

PURPLE PETAL EDU HUB

Introduction

Parallel Program Development

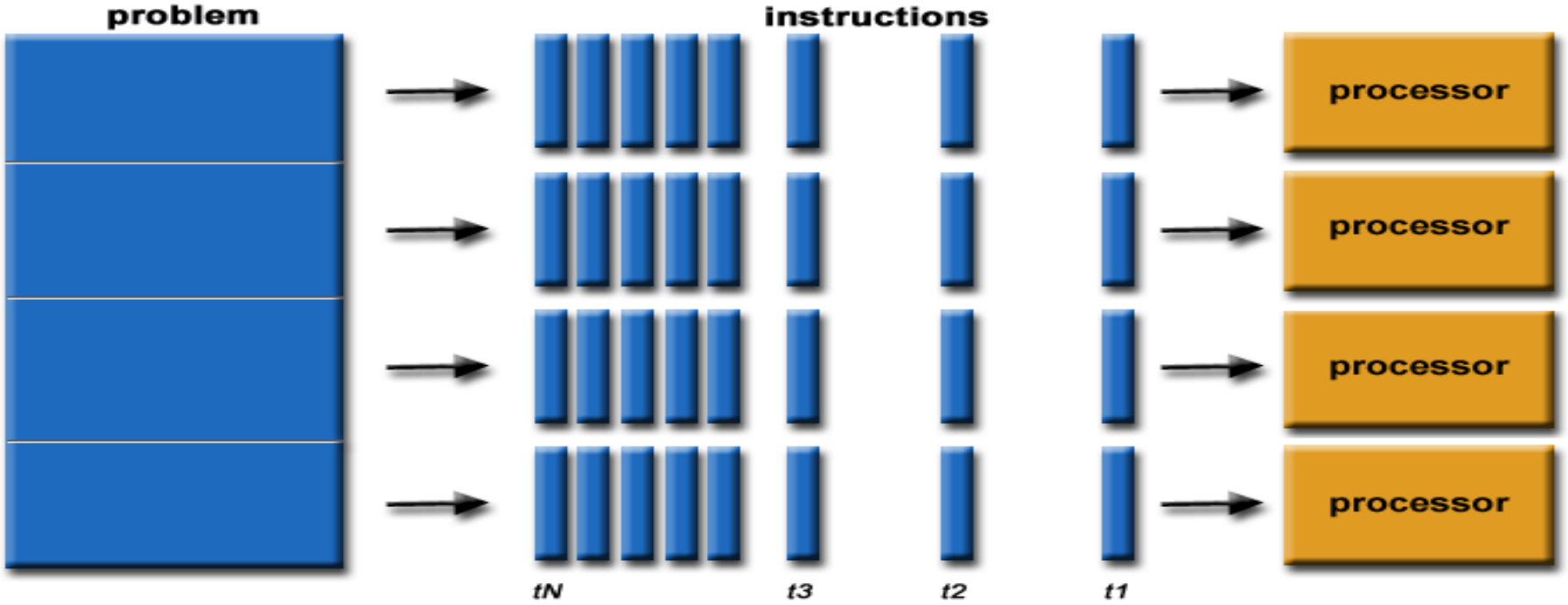
The sequence of processes is used for the development of a parallel program.The parallel computers perform the tasks parallel and in much lesser time as compared to the serial processing in the serial computers.

Serial Computing:

A problem is broken into a discrete series of instructions and these instructions are executed sequentially one after another. The problem is executed on a single processor. So, only one instruction may execute at any moment in time.

Parallel Computing:

- A problem is broken into discrete parts that can be solved concurrently and each part is further broken down to a series of instructions. These instructions from each part execute simultaneously on different processors. An overall control/coordination mechanism is employed in parallel computing.



Reasons for using Parallel Computing:

- The Real World is Massively Parallel
- Save Time And Money
- Solve Larger / More Complex Problems
- Provide Concurrency
- Make better use of underlying parallel hardware

Fields using Parallel Computing

- Science and Engineering
- Atmosphere, Earth, Environment
- Bioscience, Biotechnology, Genetics, Physics, Chemistry
- Geology, Seismology
- Mechanical Engineering
- Electrical Engineering, Circuit Design, Microelectronics
- Computer Science, Mathematics
- Defense, Weapons
- Computer simulations ... and many more

Flynn's Classical Taxonomy

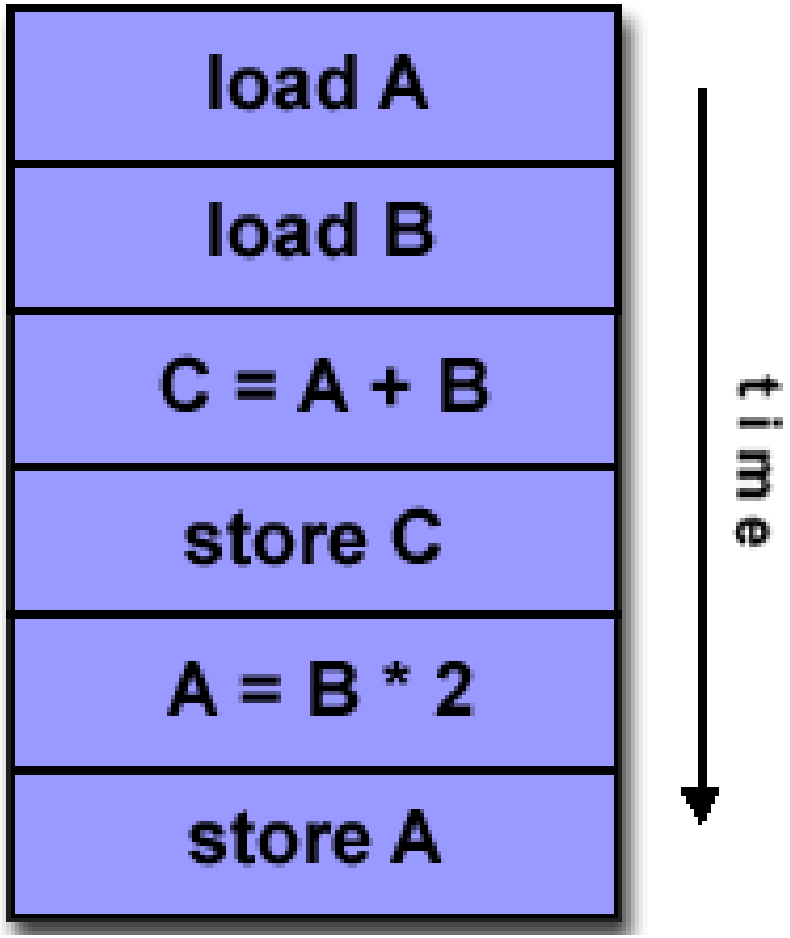
•There are different ways to classify parallel computers.

•One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.

•Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of ***Instruction Stream*** and ***Data Stream***. Each of these dimensions can have only one of two possible states: ***Single*** or ***Multiple***.

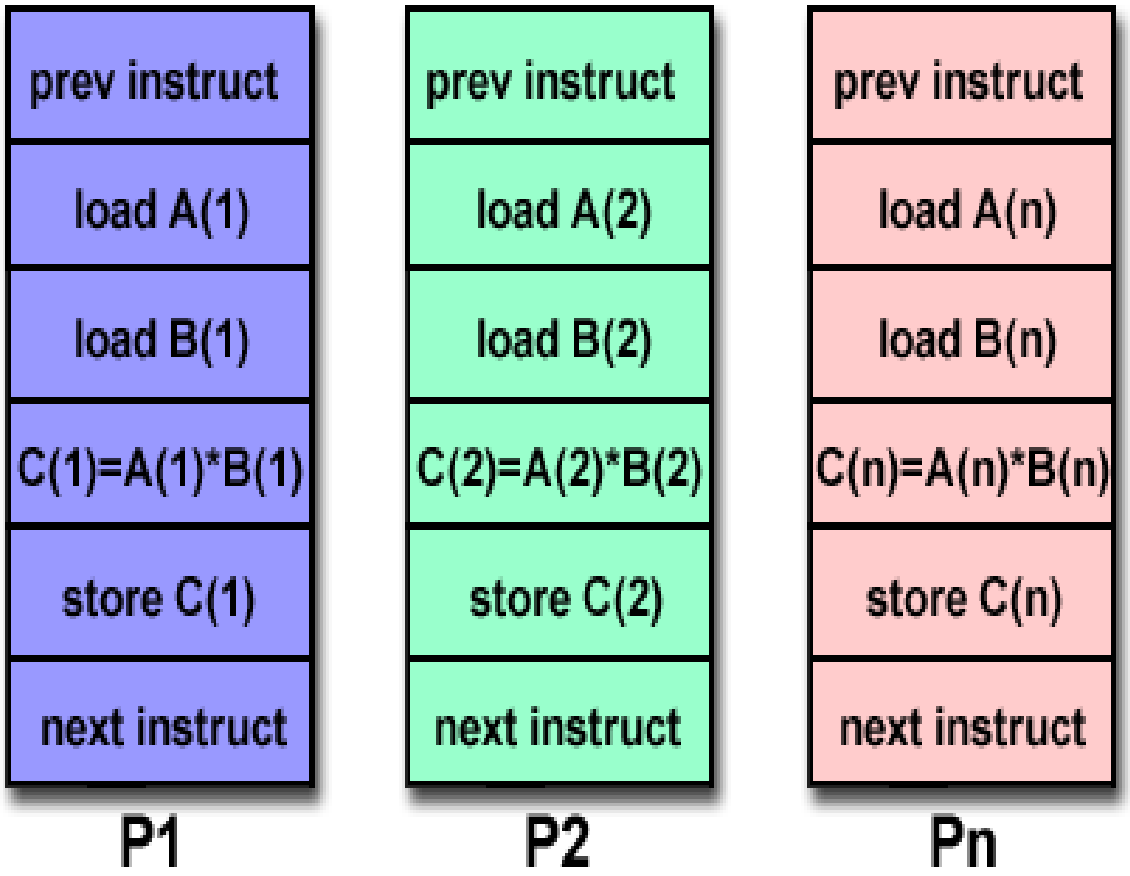
S I S D **Single Instruction Stream** **Single Data Stream**	**S I M D** **Single Instruction Stream** **Multiple Data Stream**
M I S D **Multiple Instruction Stream** **Single Data Stream**	**M I M D** **Multiple Instruction Stream** **Multiple Data Stream**

Flynn's Classical Taxonomy: SISD



- Serial
- Only one instruction and data stream is acted on during any one clock cycle

SIMD

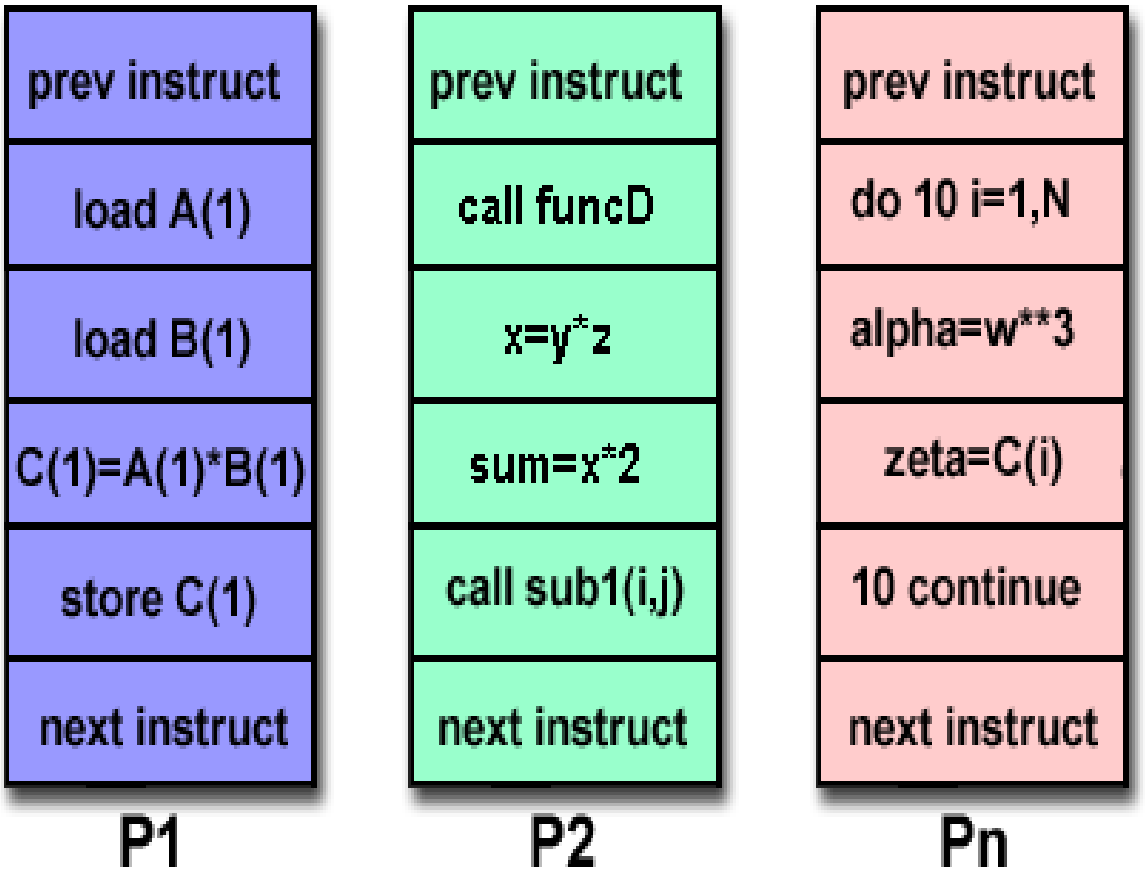


- All processing units execute the same instruction at any given clock cycle.
- Each processing unit operates on a different data element.

MISD

- Different instructions operated on a single data element.
- Very few practical uses for this type of classification.
- Example: Multiple cryptography algorithms attempting to crack a single coded message.

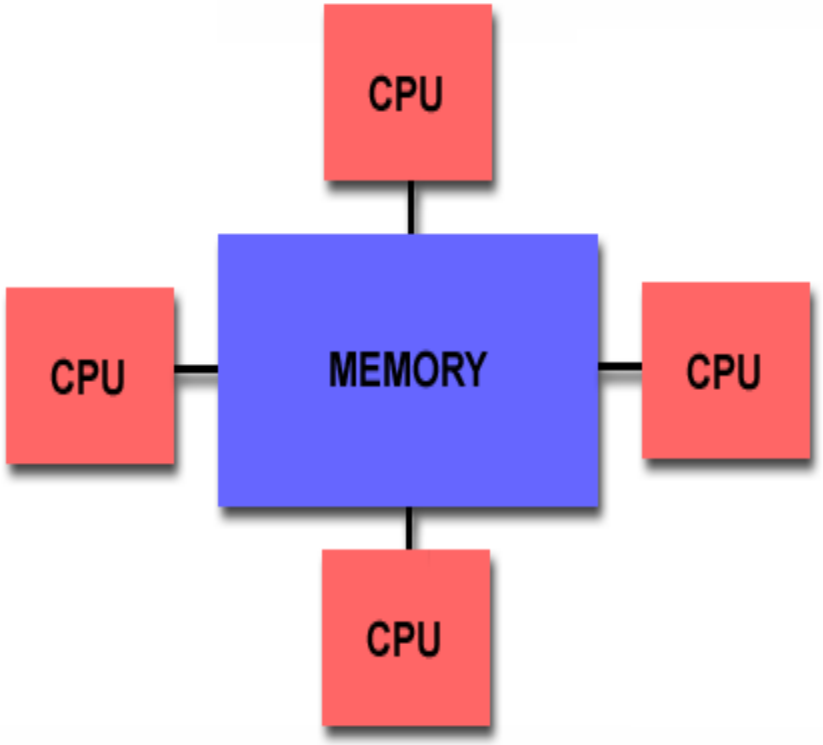
MIMD



- Can execute different instructions on different data elements.
- Most common type of parallel computer.

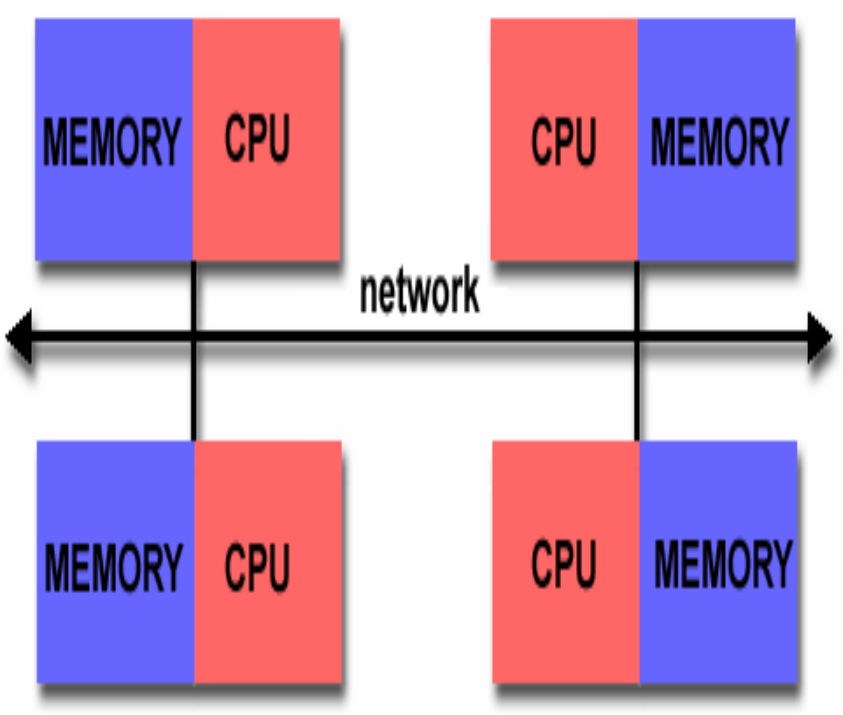
Parallel Computer Memory Architectures: Shared Memory Architecture

- All processors access all memory as a single global address space.
- Data sharing is fast.
- Lack of scalability between memory and CPUs



Parallel Computer Memory Architectures: Distributed Memory

- Each processor has its own memory.
- Is scalable.
- Programmer is responsible for many details of communication between processors.



Parallel Programming Models

- Exist as an abstraction above hardware and memory architectures
- Examples:
  - Shared Memory
  - Threads
  - Messaging Passing
  - Data Parallel

Shared Memory Model

- Appears to the user as a single shared memory, despite hardware implementations.
- Locks may be used to control shared memory access.
- Program development can be simplified since there is no need to explicitly specify communication between tasks.

Threads Model

- A single process may have multiple, concurrent execution paths.
- Typically used with a shared memory architecture.
- Programmer is responsible for determining all parallelism.

Message Passing Model

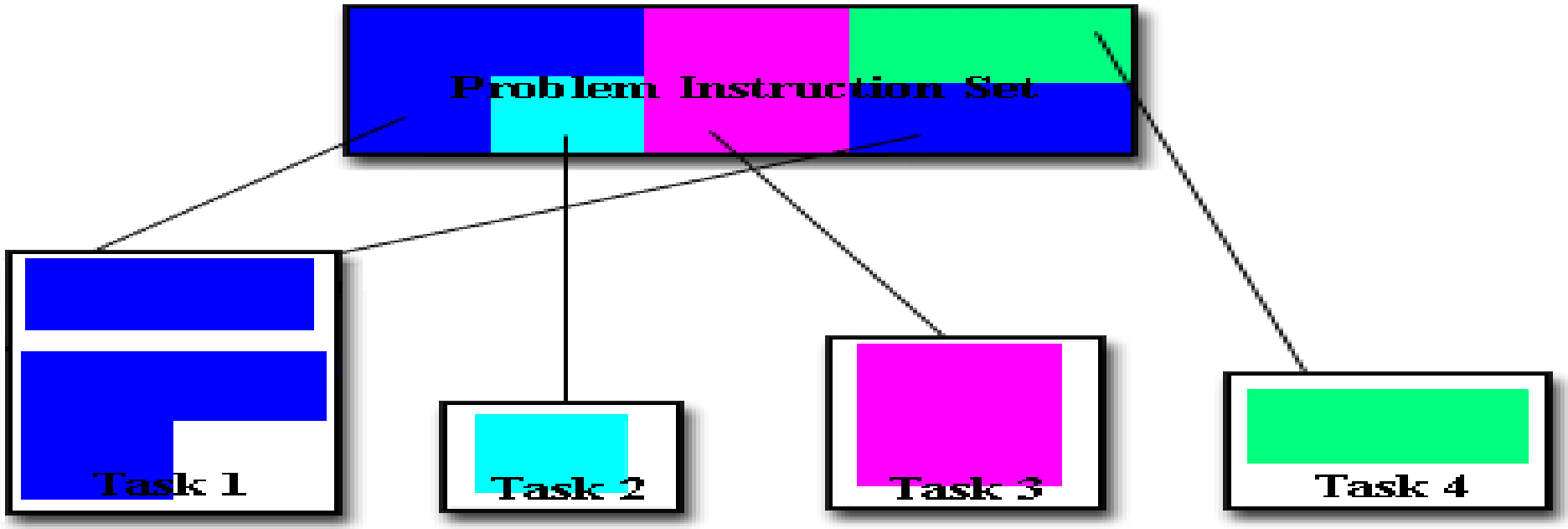
- Tasks exchange data by sending and receiving messages.
- Typically used with distributed memory architectures.
- Data transfer requires cooperative operations to be performed by each process. Ex.- a send operation must have a receive operation.
- MPI (Message Passing Interface) is the interface standard for message passing.

Data Parallel Model

- Tasks performing the same operations on a set of data. Each task working on a separate piece of the set.
- Works well with either shared memory or distributed memory architectures.

Functional Decomposition

- Each task performs a function of the overall work



Conclusion

- Parallel computing is fast.
- There are many different approaches and models of parallel computing.
- Parallel computing is the future of computing.