**Parallel computing**

**Dr Ani Brown Mary N, Dr Anne Wincy, Dr Abbiramy V S, Dr Shameem Fathima A**

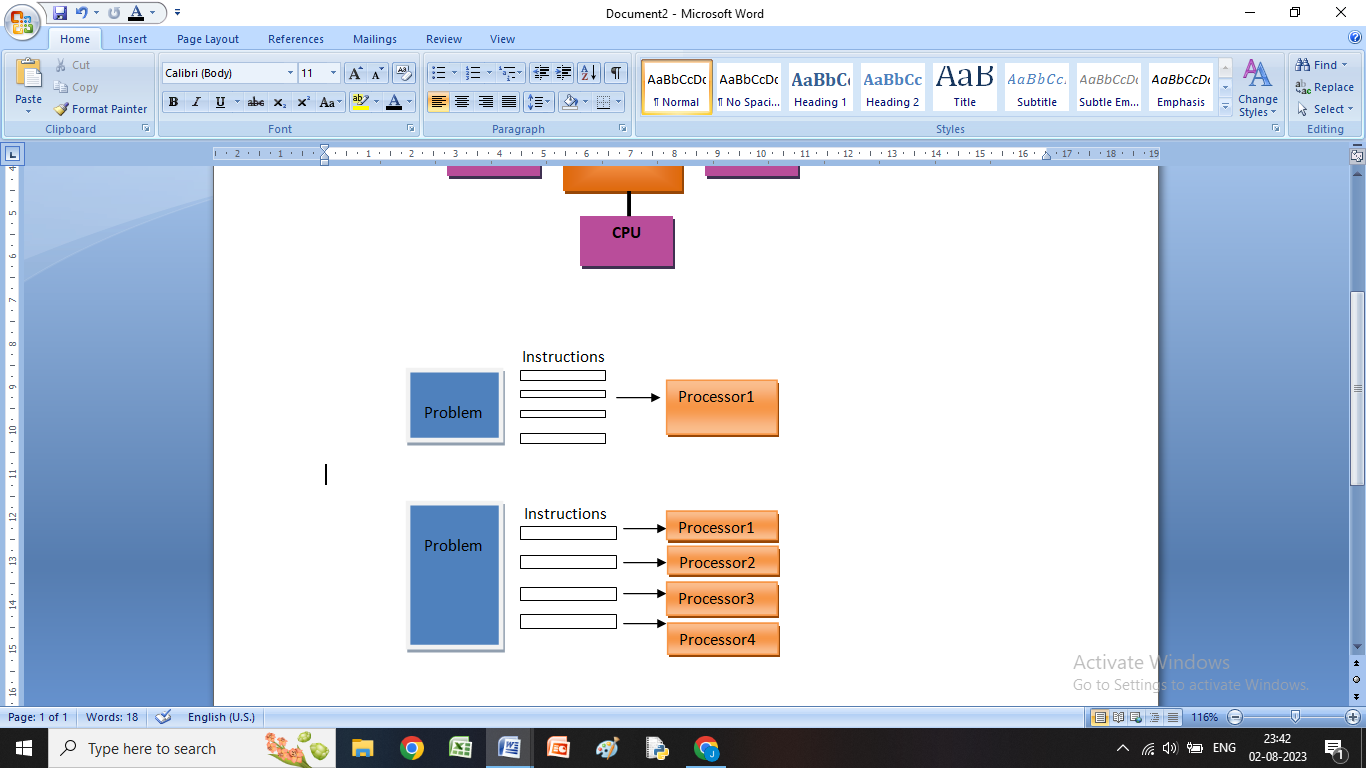
**Assistant Professor, Department of Computer Science**

**Sarah Tucker college**

**Corresponding Author emailID:** [**anibrownvimal@gmail.com**](mailto:anibrownvimal@gmail.com)

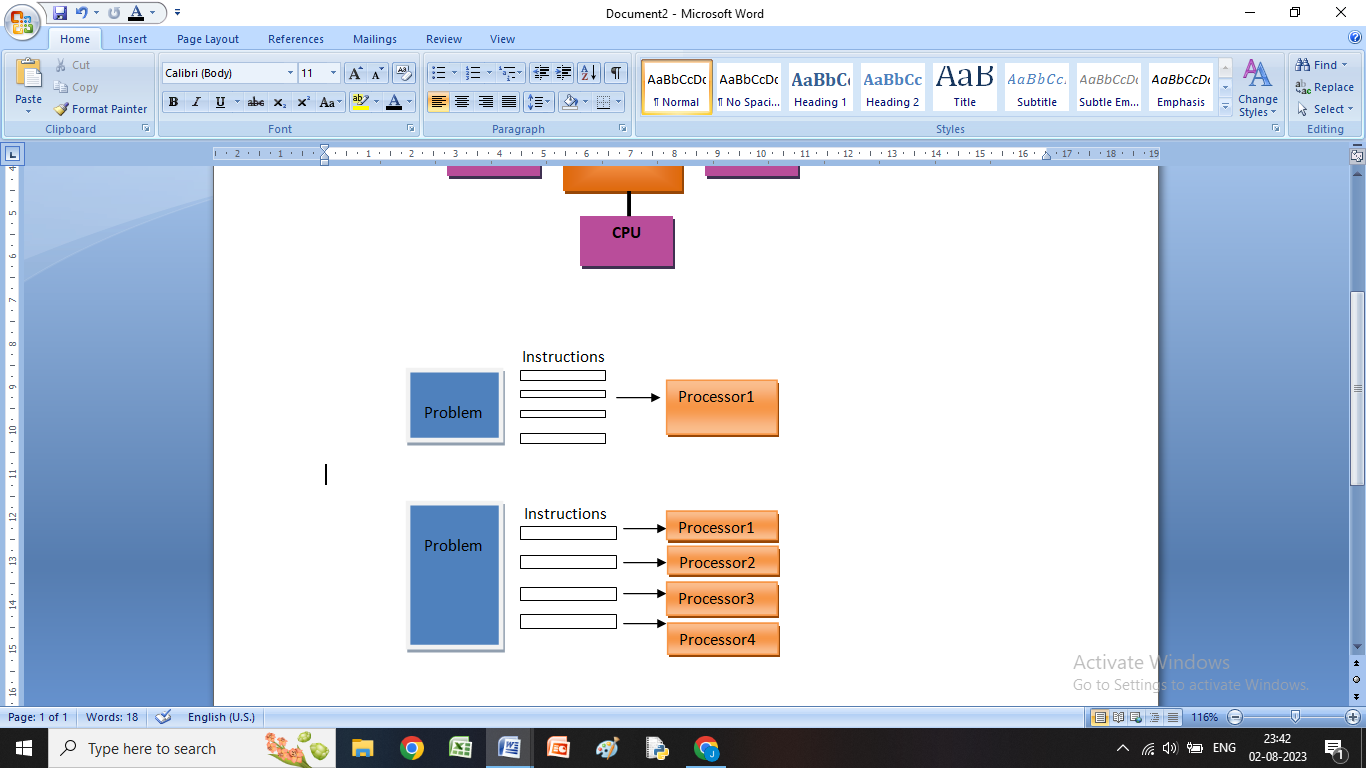
**I Introduction**

Parallel computing refers to a group of processors working simultaneously to solve a large computational problem. Splitting of task into subtasks on multiple processors helps to obtain the results faster [1]. It is done by multiple processors communicating through shared memory, which combines the results after completion. High performance computing can be achieved by parallel computing by throwing more resources at a task will reduce its time to completion, with potential cost savings[2]. In traditional Serial Computing, a problem is broken into a smaller series of instructions. Each instruction is executed sequentially by a single processor one instruction at a time [3] as shown in the Figure 1.



**Figure 1 Serial Computing**

#### Parallel computing uses several processors simultaneously to execute multiple calculations. A problem is broken down into smaller units. Instructions from each part are executed simultaneously on different processors at the same time [4]. Large Complex problem can be solved in less time with multiple compute resources than with a single compute resource. Typically parallel computing infrastructure is kept in a single datacenter, where many processors are located in a server rack. The application server distributes compute requests in small pieces, which are subsequently executed concurrently on each processor as shown in the Figure 2.



**Figure 2 Parallel Computing**

Parallel computing can solve larger problems in a short time, when compared with serial computing. In the case of simulation, modeling and understanding complex real-world fact,parallel computing is much suitable while comparing with serial computing. Parallel computing has made a great impact on a variety of areas from computational simulations for Engineering and scientific applications to commercial applications in data mining and transaction processing. The main reasons for using parallel computing are save time, solve larger problems and to provide concurrency. The benefit also includes cost savings, overcoming memory constraints and takes the merits of nonlocal resources. The aim of parallel computing is to increase the computation power for application processing to be faster and solving the problem. Parallel computing is considered to be the high end computing and applied to model difficult problems in many areas of science and engineering like Biotechnology, Genetics, nuclear, Geology, Computer science, Defense, weapons etc.

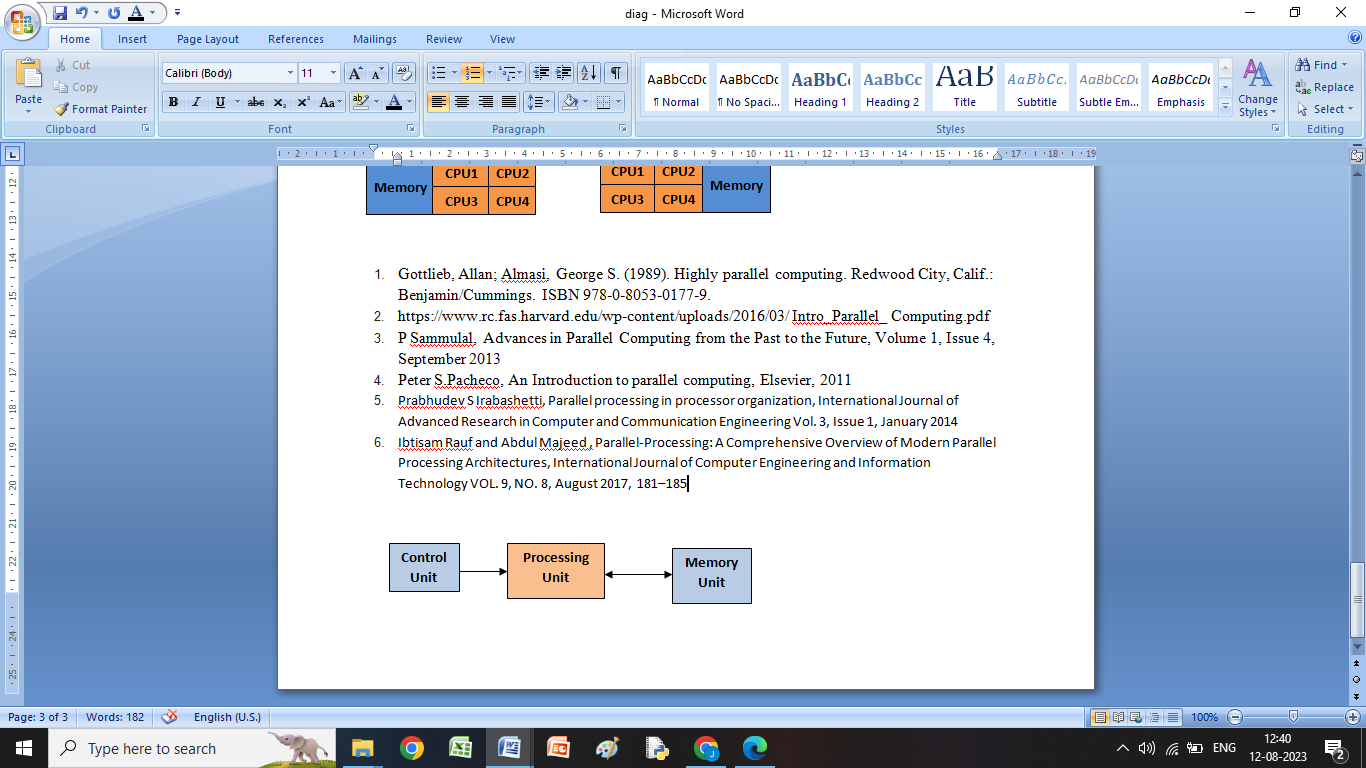
In the past 10 years, the development indicated by ever faster networks, distributed systems, and multi-processor computer architectures suggest that parallelism is the future of computing. Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently [11]. In fine-grained parallelism, a program is divided into a large number of small subprograms. These subprograms are assigned individually to each of the available processors. The amount of work carried out is low but the work is evenly distributed among the processors. Hence, fine-grained parallelism achieves Load balancing. Speedup is defined as the ratio of the execution time of a task on a larger machine to the execution time of the same task on the machine. Handling larger tasks by increasing the degree of parallelism is called scale up. It is the form of parallel computing which is based on the increasing processor’s size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data [13]. A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism [12].

**II Processor organization**

Parallel Systems are systems designed with multiple processors to execute the processes. Hence many operations can be carried out simultaneously resulting in improved processing and I/O speed. Parallel Computer Architecture is the method of organizing all the resources to maximize the performance and the programmability within the limits given by technology and the cost at any instance of time. It adds a new dimension in the development of computer system by using more and more number of processors. For parallel computers and programs, Michael J. Flynn introduced one of the earliest classification systems, which is known as Flynn's taxonomy [4]. The categorization for the organization of a computer system is based on the number of instructions and data items that are manipulated simultaneously.

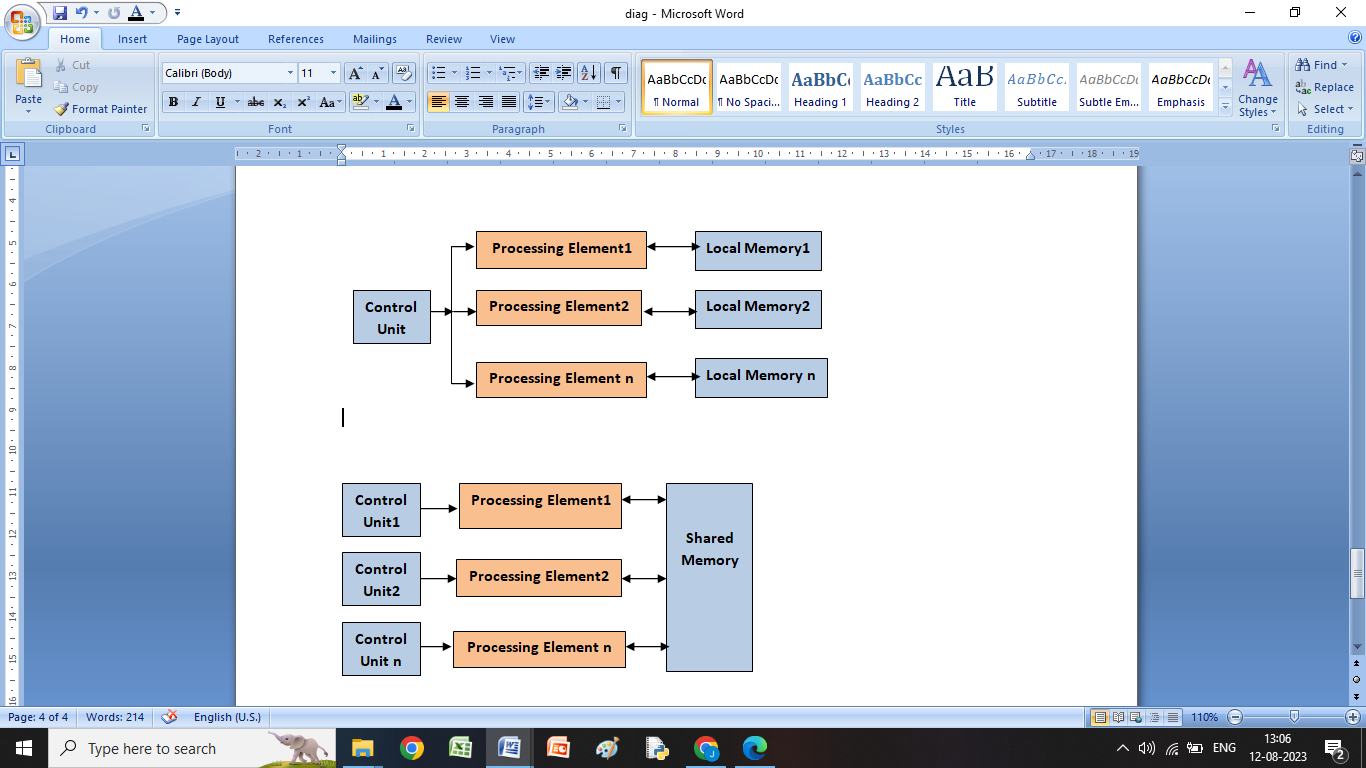
1. Single instruction stream, single data stream (SISD)
2. Single instruction stream, multiple data stream (SIMD)
3. Multiple instruction stream, single data stream (MISD)
4. Multiple instruction stream, multiple data stream (MIMD)

**A Single instruction stream, Single Data stream (SISD)** refers to a computer architecture in which a uniprocessor, executes instructions sequentially. SISD [5] can have concurrent processing characteristics. In modern SISD computer, Instruction fetching and pipelined instruction execution is the examples as shown in the Figure 3. It represents the organization of a single computer containing a control unit, a processor unit, and a memory unit. Instructions are executed sequentially, and the system may or may not have internal parallel processing capabilities.



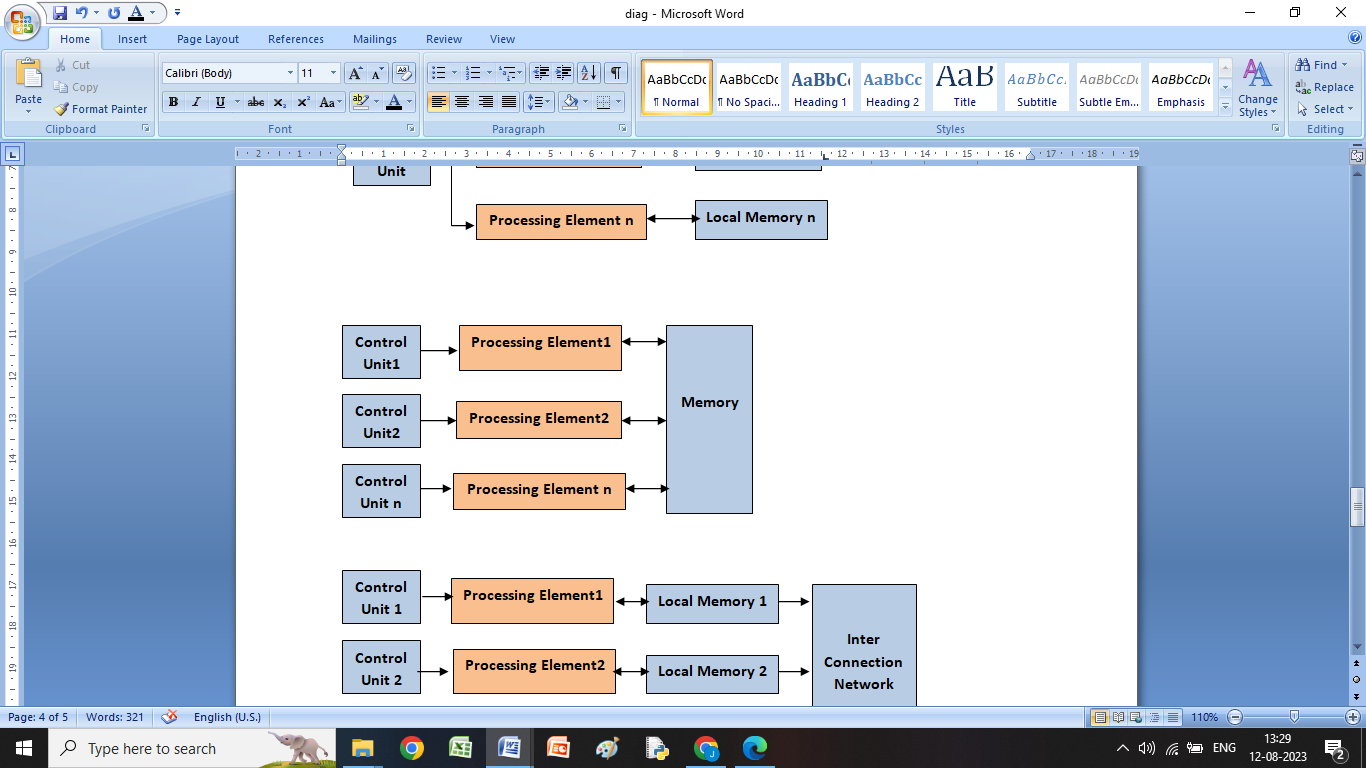
**Figure 3 SISD**

**B Single Instruction stream Multiple Data stream** **(SIMD)** has multiple processing elements. It contains computers with multiple processing elements that perform the same operation on multiple data points simultaneously. Thus, such machines utilize data level parallelism. SIMD is used in some common tasks like adjusting the contrast of image or adjusting the volume of digital audio. Most designs of CPU include SIMD instructions to improve the performance of multimedia. All processors in a parallel computer execute the same instructions but operate on different data at the same time.



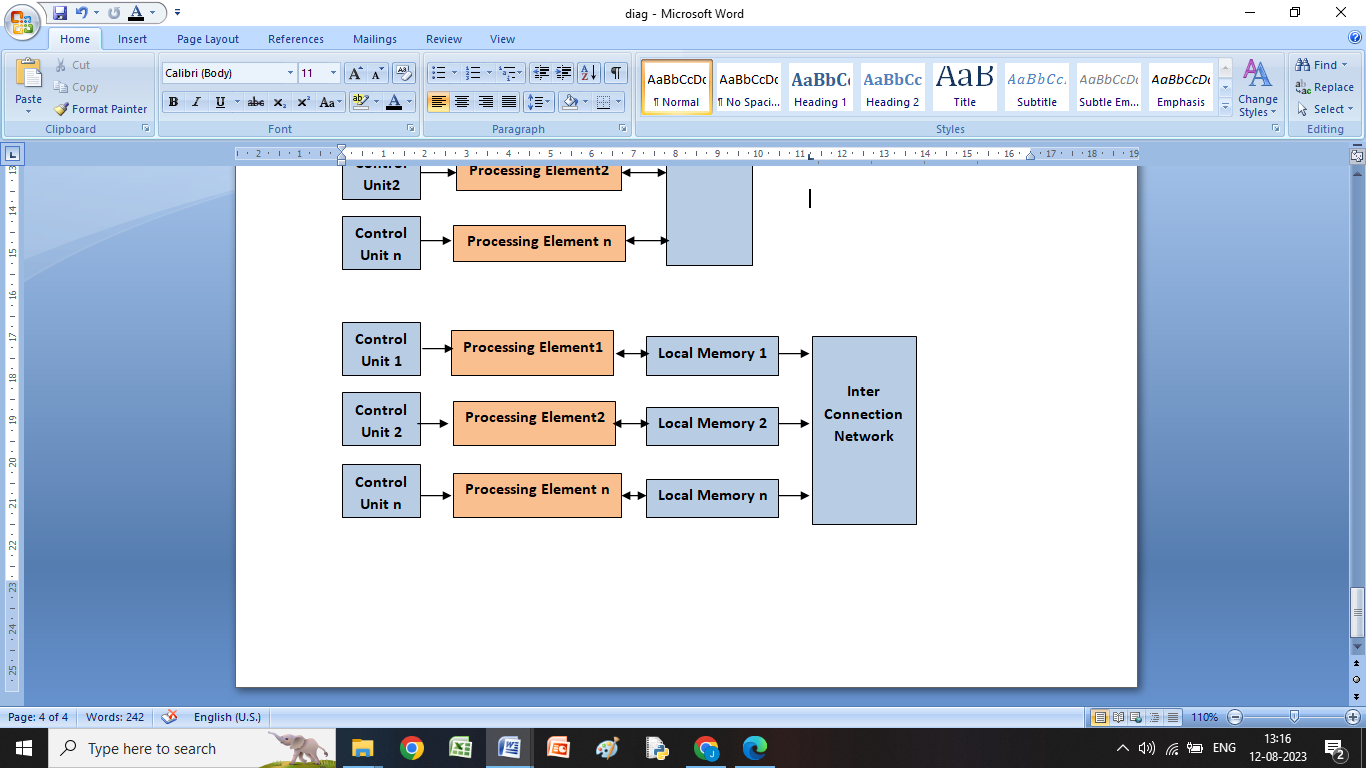
**Figure 4 SIMD**

**C Multiple instruction stream, single data stream (MISD)** is a type of parallel computing architecture where multiple processing units perform different operations on the same data. The result of a processing unit becomes the input of the next processor. Such machines utilize Pipeline architecture. Fault-tolerant computers normally execute the same instructions repeatedly in order to detect and mask errors, known as task replication, may be belongs to this type. Example of MISD in computing is the Space Shuttle flight control computers. MISD is a type of parallel computing architecture where multiple processing units process only a single data stream, but each processing unit works independently on different sets of instruction streams belonging to that one data stream.



**Figure 5 MISD**

**D Multiple instruction stream, multiple data stream (MIMD) [6]** is a technique applied to achieve parallelism. Machines using MIMD have different processors can execute different instructions on various data at the same time. MIMD is used many application areas such as computer-aided design, simulation, modeling, and communication switches. MIMD machines can have either shared memory or distributed memory. A multi-core CPU is a example of MIMD machine. Machines using MIMD have a number of processors that function asynchronously and independently. All processors in a parallel computer can execute different instructions and operate on different data at the same time. Parallelism achieved by connecting multiple processors together.



**Figure 6 MIMD**

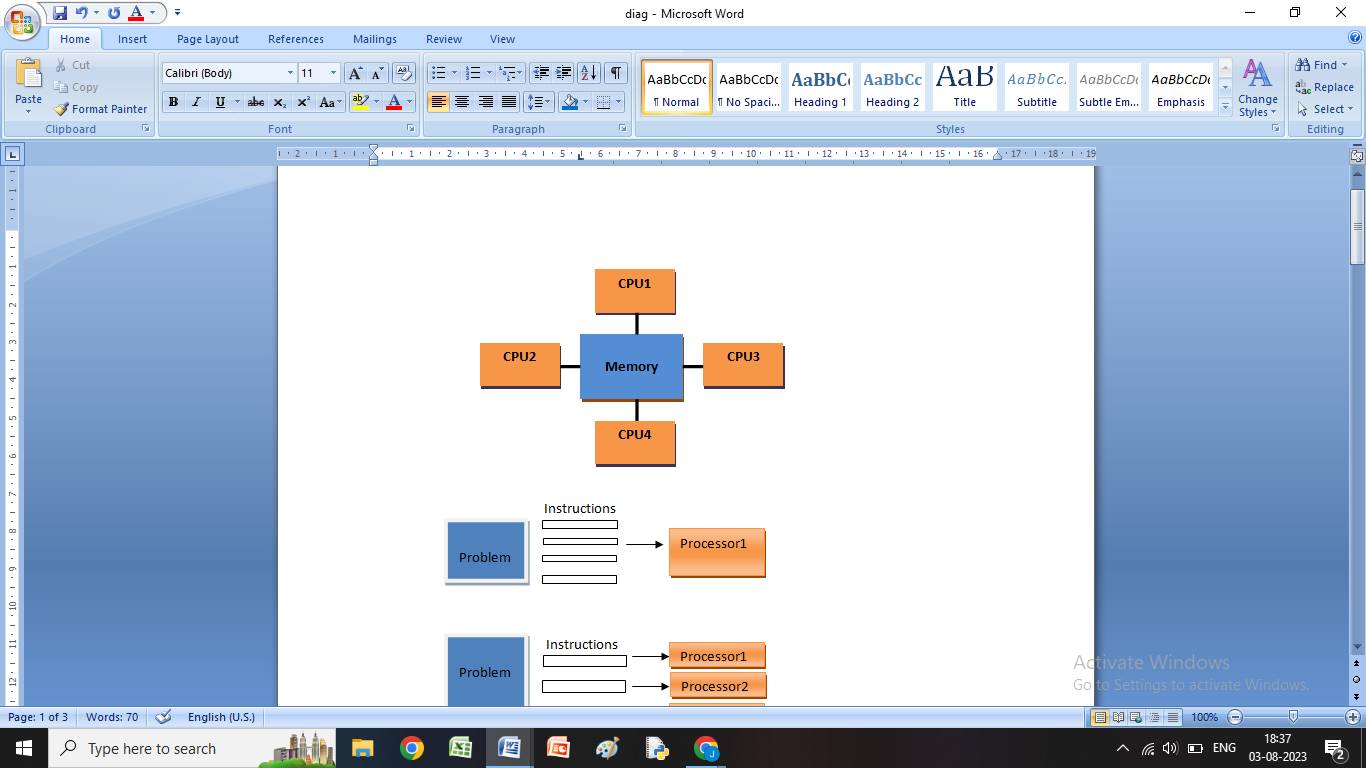
Concurrent systems are systems that are built from a set of independent components which perform their operations at the same instant of time. These components interact in a controlled manner so that the interactions do not produce unwanted issues such as deadlock and resource starvation. These components or processes are interleaved on the processor and the available resource is shared among the processes [9]. In coarse-grained parallelism, a program is divided into large subprograms. When assigned to processors, it takes large amount of computation time. This might result in load imbalance among the processors. It also fails to achieve parallelism as most of the computation is performed sequentially on a processor. The advantage of this type of parallelism is low communication and synchronization overhead [10].

**III Shared memory**

MIMD uses shared memory [7] which generally has the ability to access all memory as global address space. Multiple processors can operate independently but share the same memory resources. Any changes done in memory by any processor is visible to all other processors. Based on memory access time, shared memory machines can be classified as Uniform Memory Access and Non-uniform Memory Access.

**A** **Uniform Memory Access (UMA)**

Identical processors share the physical memory uniformly. All the processors have equal access time to memory which is represented by Symmetric Multiprocessor (SMP) machines. Sometimes known as cache coherent UMA, if any processor updates a location in shared memory and the update is communicated to all processors through cache memory.



**Figure 7 UMA**

### B Non-uniform Memory Access (NUMA)

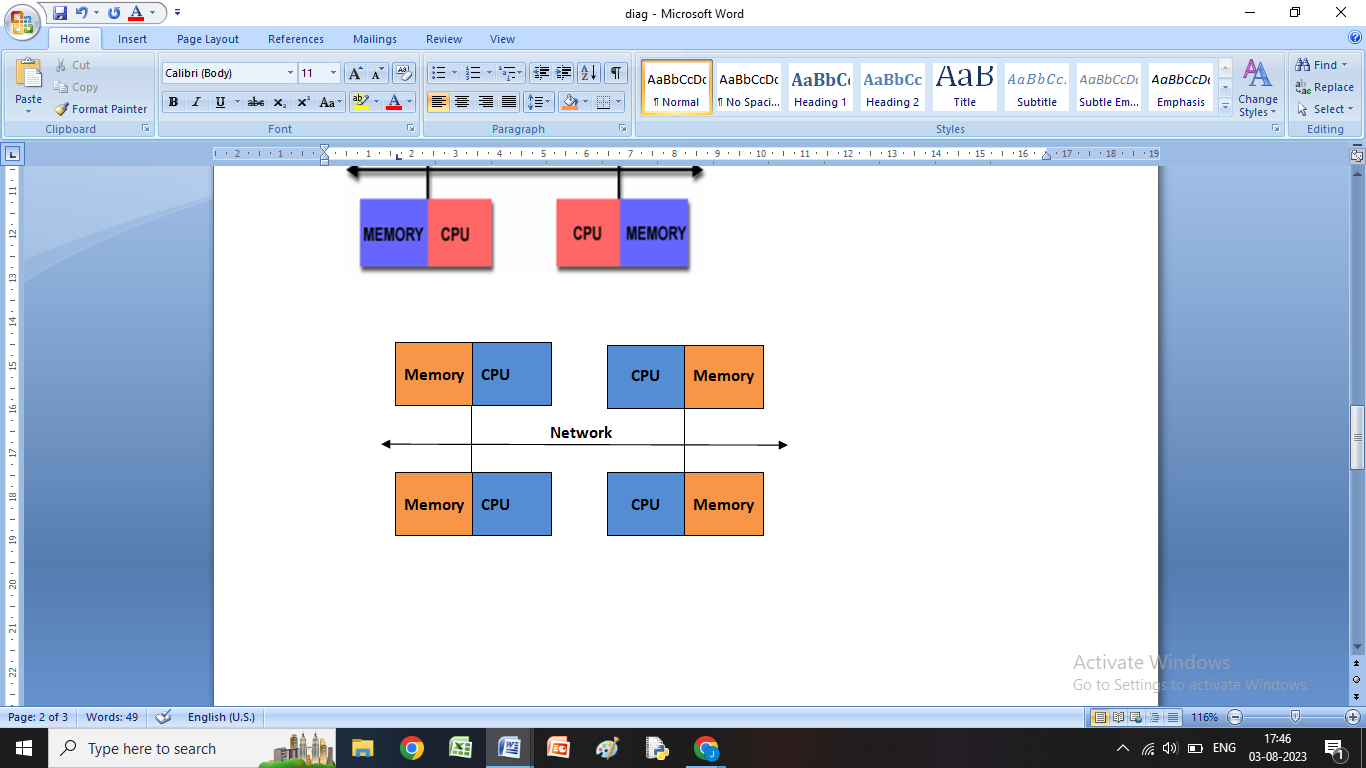
### NUMA physically links two or more Symmetric Multiprocessors. All processors have equal access to all memories. But the access time of processor to all memories differs. If cache coherency is maintained, then it may be also called as Cache Coherent NUMA. Shared memory is user-friendly programming perspective due to Global address space and fast data sharing.

### 

### Figure 8 NUMA

**C Distributed memory**

MIMD uses Distributed memory [8], which requires a communication network to connect inter-processor memory. Processors have their own local memory. So it operates independently. Changes in local memory have no effect on the memory of other processors. It also requires communication to exchange data among processors. Synchronization between tasks is necessary. It benefits rapid access to its own memory without any Interference and overheads of cache coherency.  Memory is scalable with number of processors.



**Figure 9 Distributed Memory**

If a group of asynchronously executing processors, share access to same data structure in an unstructured way errors occur. Two processors may end up using same prime value to sieve through array location and finds it unmarked[14].

**IV Computational Demands of Contemporary Science**

As someone working on problems in computer science in the 1990s, I often used to be asked: What languages do you work in? I would naturally answer, deliberately misunderstanding the question: Mostly in English, sometimes also in Tamil. In those days, working with computers intended writing programs in Fortran or Cobol or C, and that was what the questioner was asking about [15]. My answer was about the programming language being irrelevant, the underlying concepts being more important. In fact, a more precise but entirely murky answer would have been first order logic, and to a lesser extent, algebra, these being the languages for abstract reasoning about computation [16].

All this is to point out that the public perception of computing and computer science may not reflect the thinking that underlies these disciplinary domains. (This is quite natural; the public insight of methods used by electrical engineers or archaeologists is doubtful to be perfect either.) The increasing impact of computers on modern living is not necessarily a reason to expect such understanding either: people consult doctors all the time but do not expect to understand medical diagnosis and prescription. It is when there is advocacy of such ​“disciplinary thought” in school education that it becomes important to examine such thought, and when it comes to school education, public perception and engagement is critical [17].

Computers have become indispensable to scientific research. They are essential for collecting and analyzing experimental data, and they have largely replaced pencil and paper as the theorist’s main tool. Computers let theorists extend their studies of physical, chemical, and biological systems by solving difficult nonlinear problems in magneto hydrodynamics; atomic, molecular, and nuclear structure; fluid turbulence; shock hydrodynamics; and cosmological structure formation [18]. Beyond such well-established aids to theorists and experimenters, the exponential growth of computer power is now launching the new field of computational science [19]. Multidisciplinary computational teams are beginning to develop large-scale predictive simulations of highly complex technical problems [20].

While this has generated quite a buzz in the country, it is quite unclear whether there is a clear perception among the community of educators and teachers what CT is about, why it is being coupled with mathematical thinking at all, and whether promoting CT in schools is necessary or even desirable. The coupling of mathematical and computational thinking is significant, since this suggests completely doing away with the current model of ​“computer classes” and moving over to teaching the science underlying computing, the emphasis being on thinking. This has important implication for mathematics education as well, shifting the focus from learning “operations”, formulas and procedures (to solve equations, etc.) to learning a way of thinking [21].

Modern science demands high-performance computing platforms that support a diverse range of activities, from quantum mechanical simulations of high-performance electronic materials, large scale molecular dynamics simulations, to data-driven machine learning-based analysis of high-resolution, high-content scientific images [22]. On the other hand, the justification for doing so, according to the passage, stems from the importance of mathematics and computational thinking for ​“upcoming fields and professions such as artificial intelligence, machine learning, and data science, etc.” The field has reached a threshold at which better organization becomes crucial [23]. New methods of verifying and validating complex codes are mandatory if computational science is to fulfill its promise for science and society [24].

**V Conclusion**

This chapter provides a detailed explanation on designing, analyzing, and implementing parallel algorithms for computers that have numerous processors. Many algorithms suitable for conventional, single -processor computers are not suitable for parallel architectures. Many algorithms with inherent parallelism have a higher computational complexity than the best sequential counterpart. For decades computer architects have incorporated parallelism into various levels of hardware in order to increase the performance of computer systems. To achieve the extremely high speeds demanded by contemporary science, architectures must now incorporate parallelism at the highest levels of the system. The fastest computers in the world use high level parallelism. These systems are leading to new scientific discoveries.

**References**

1. Gottlieb, Allan; Almasi, George S. (1989). Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings. ISBN 978-0-8053-0177-9.
2. https://www.rc.fas.harvard.edu/wp-content/uploads/2016/03/ Intro\_Parallel\_ Computing.pdf
3. P Sammulal, Advances in Parallel Computing from the Past to the Future, Volume 1, Issue 4, September 2013
4. Peter S.Pacheco, An Introduction to parallel computing, Elsevier, 2011
5. Prabhudev S Irabashetti, Parallel processing in processor organization, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 1, January 2014
6. Ibtisam Rauf and Abdul Majeed, Parallel-Processing: A Comprehensive Overview of Modern Parallel Processing Architectures, International Journal of Computer Engineering and Information Technology VOL. 9, NO. 8, August 2017, 181–185
7. https://sites.engineering.ucsb.edu/~hpscicom/p1.pdf
8. O. Serlin, The Serlin Report On Parallel Processing, No.54, pp. 8-13, November 1991.
9. <https://www.oreilly.com/library/view/foundations-of-scalable/9781098106058/ch04.html>
10. Yeung, Donald; Dally, William J.; Agarwal, Anant. "How to Choose the Grain Size of a Parallel Computer". CiteSeerX 10.1.1.66.3298.
11. Quinn, Michael J. (2007). Parallel programming in C with MPI and openMP (Tata McGraw-Hill ed.). New Delhi: Tata McGraw-Hill Pub. ISBN 978-0070582019.
12. <https://www.scribd.com/doc/33700101/Instruction-Level-Parallelism#scribd>
13. David E. Culler, Jaswinder Pal Singh, Anoop Gupta. Parallel Computer Architecture - A Hardware/Software Approach. Morgan Kaufmann Publishers, 1999. ISBN 1-55860-343-3, pg 15.
14. von Eicken T, Culler D, Goldstein S and Schauser K 1992 Active messages: A mechanism for integrated communication and computation Proc. 19th International Symp. Computer Arch. (Queensland, Australia)
15. Lumsdaine A, Gregor D, Hendrickson B and Berry J 2007 Parallel Processing Letters 17 5–20
16. Bengio Y, Glorot X, Understanding the difficulty of training deep feedforward neural networks. Paper presented at: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics; 2010 May 13–15; Chia Laguna Resort, Sardinia, Italy. p. 249–256.
17. Wing, Jeannette, “Computational thinking and thinking about computing.” Philosophical Transactions of the Royal Society, 366 (1881), 3717 – 3725, 2008.
18. Denning, Peter J. “The science in computer science.” Communications of the ACM, 56(5), 35 – 38, 2013.
19. Computational Social Science and the Study of Political Communication Yannis Theocharis and Andreas Jungherr
20. POLITICAL COMMUNICATION 2021, VOL. 38, NOS. 1–2, 1–22 https://doi.org/10.1080/10584609.2020.1833121
21. King, G., & Persily, N. (2019). A new model for industry–academic partnerships. PS, Political Science & Politics. https://doi.org/10.1017/S1049096519001021
22. Kuklinski, J. H., Quirk, P. J., Jerit, J., Schwieder, D., & Rich, R. F. (2000). Misinformation and the currency of democratic citizenship. The Journal of Politics, 62(3), 790–816. https://doi.org/10. 1111/0022-3816.00033
23. Lazer, D., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S. A., Sunstein, C. R., Thorson, E. A., Watts, D. J., & Zittrain, J. L. (2018). The science of fake news. Science, 359 (6380), 1094–1096. https://doi.org/10.1126/science.aao2998
24. Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., Van Alstyne, M., & Christakis, N. (2009). Social science: Computational social science. Science, 323(5915), 721–723. https://doi.org/10. 1126/science.1167742