UNIT: 1

Topics to be covered:

1. Introduction
2. Flynn's Classification
3. System Attributes to Performance
4. Parallel Computer Models
    - Multiprocessors and Multicomputer
    - Multifactor and SIMD Computers
5. Data and Resource Dependences
6. Hardware and Software Parallelism
7. Program Partitioning and Scheduling
8. Grain Size and Latency
9. Control Flow and Data Flow
10. Demand Driven Mechanism
11. Static Interconnection Networks
12. Dynamic Interconnection Networks
    - Bus system
    - Crossbar Switch
    - Multiport Memory
    - Multistage and Combining Networks

Flynn's Classification:-

Computer Organizations are characterized by the multiplicity of the hardware provided to service the instruction and data streams. Four machine organizations are as follows:-

1. Single Instruction Stream and Single Data Stream(SISD)
2. Single Instruction Stream and Multiple Data Stream(SIMD)
3. Multiple Instruction Stream and Single Data Stream(MISD)
4. Multiple Instruction Stream and Multiple Data Stream(MIMD)

SISD Computer Organization:-

- It represents most serial computers available today.
- Instructions are executed sequentially but may overlap in their execution stages (pipelining).
- Most SISD uniprocessor are pipelined.
- An SISD computer may have more than one functional unit in it.
- All the functional units are under the supervision of one control unit.

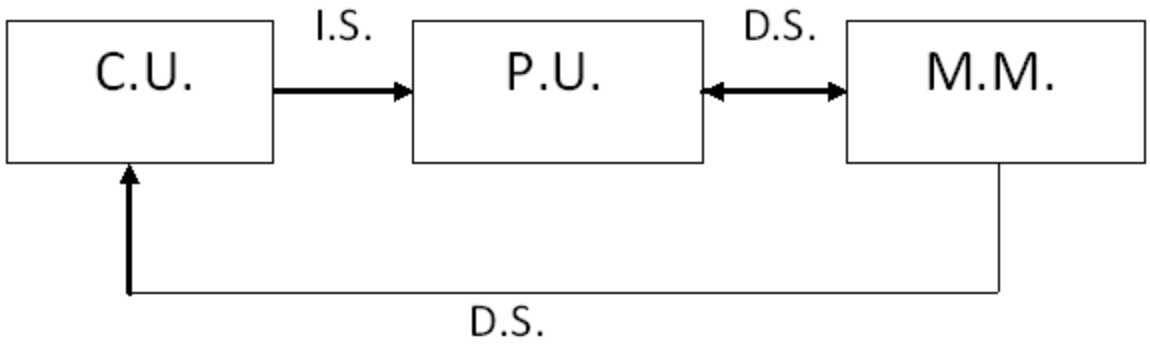


FIG:-SISD COMPUTER ORGANIZATION

C.U. – Control Unit P.U. - Processor Unit

I.S. - Instruction Set D.S. – Data Set

SIMD Computer Organization:-

· This corresponds to array processors.
· There are multiple processing elements supervised by the same control unit.
· All PE's receive the same instruction broadcast from the control unit but operate on different data set from modules.

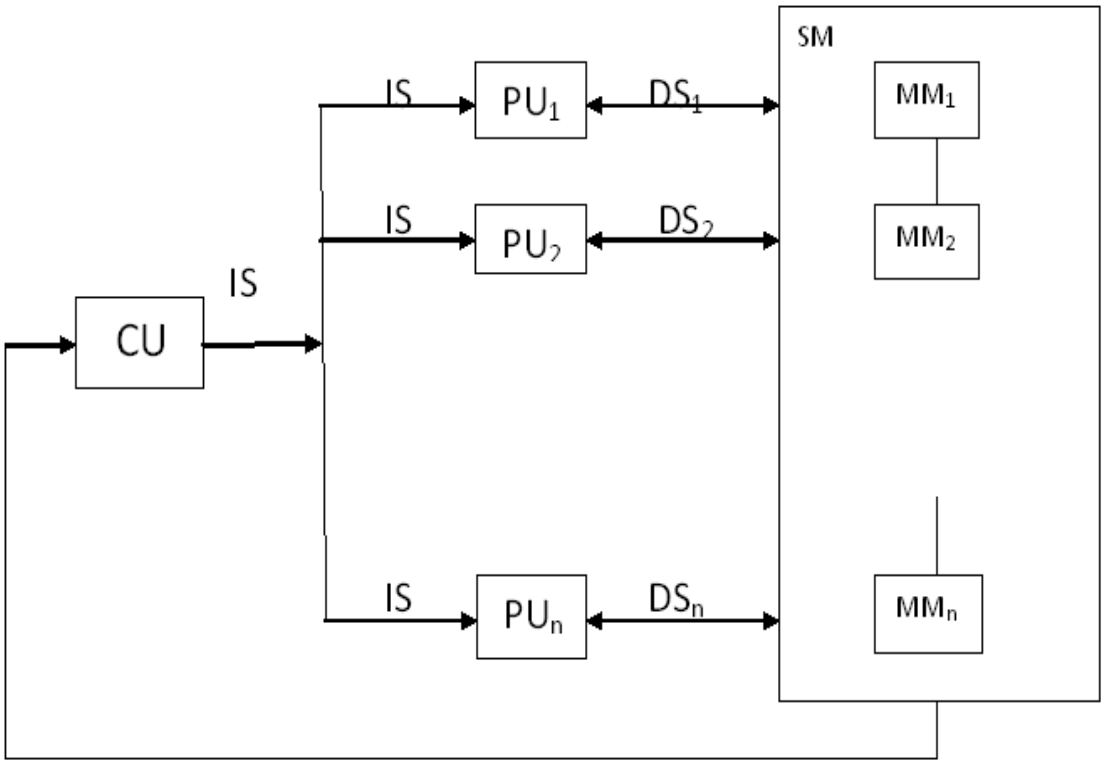


FIG:- SIMD COMPUTER ORGANIZATION

MISD Computer Organization:-

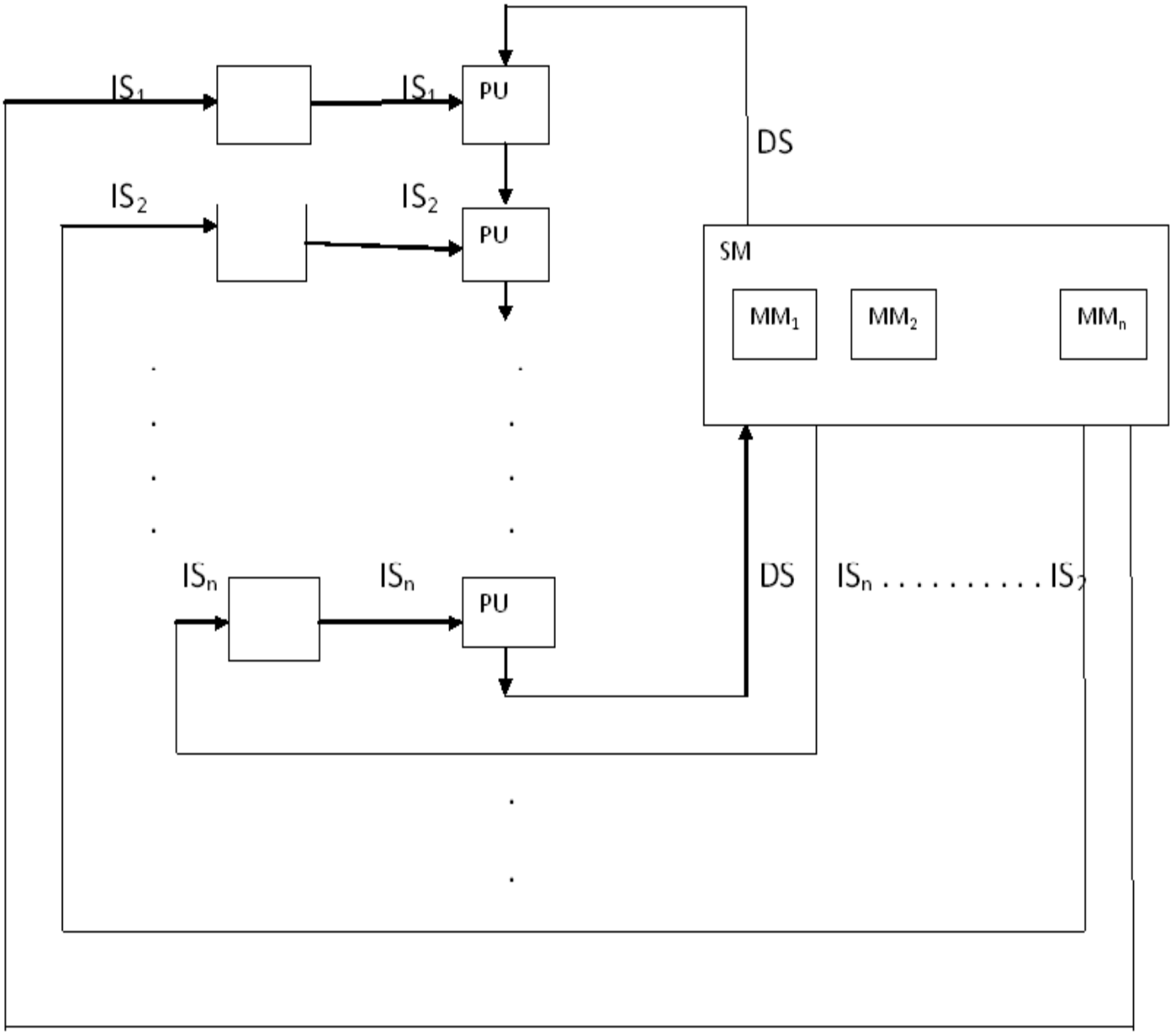
· There are n processor units, each receiving distinct instructions

operating over the same data stream and its derivatives.

· The results (output) of one processor become the input (operands) of the next processor in the macro pipe.

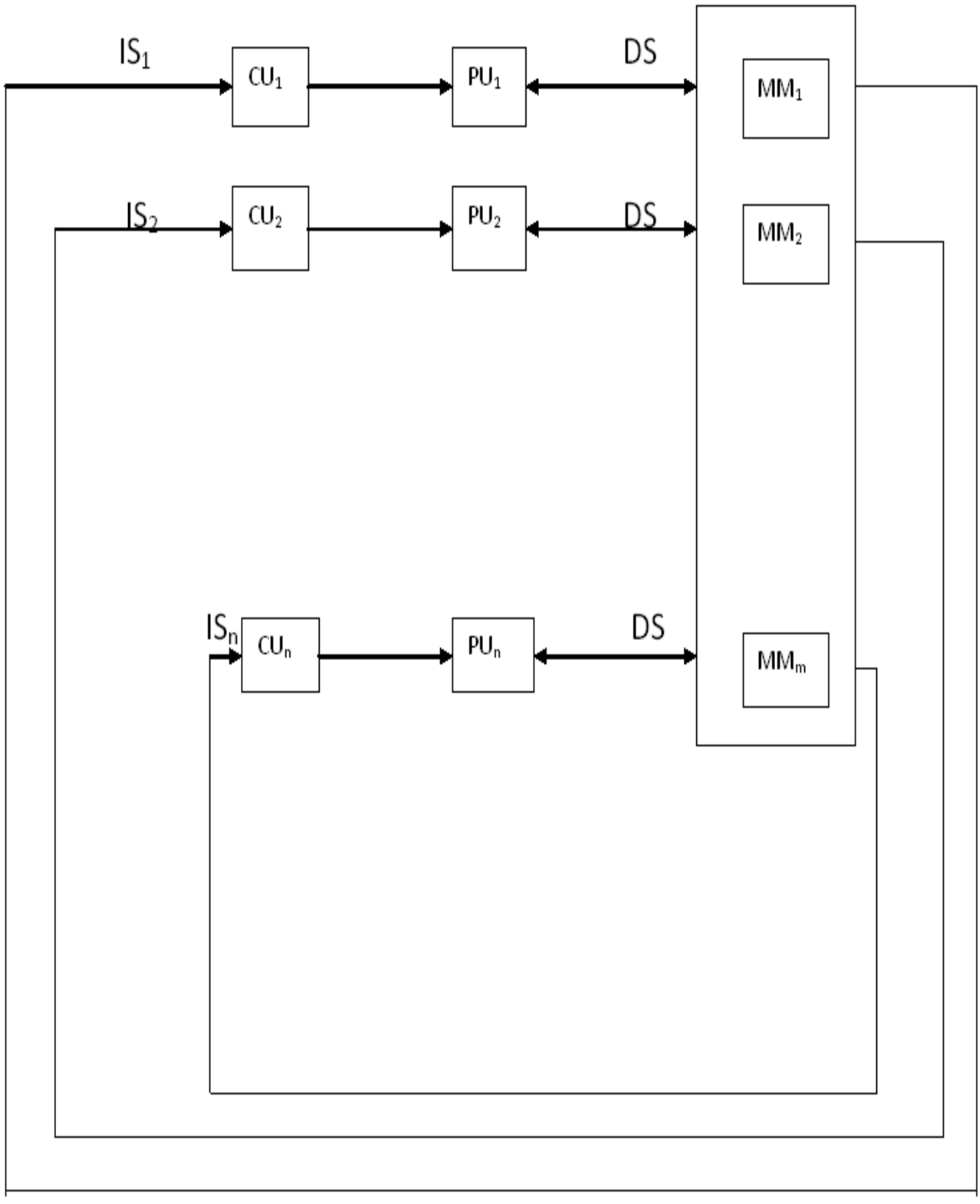
· This structure has received much less attention and has been challenged as impractical by some computer architects.

· No real embodiment of this class exists.



MIMD Computer Organization:-

- Most multiprocessor systems and multiple computer systems can be classified in this category.
- An intrinsic MIMD Computer implies interactions among the n processors because all memory streams are derived from the same data space shared by all the processors.
- If the n data streams were derived from disjointed subspaces of the shared memories then we would have the so called multiple SISD (MSISD) operation, which is nothing but a set of n independent SISD uniprocessor system.
- An intrinsic MIMD computer is tightly coupled if the degree of interactions among the processors is high. Otherwise, we consider them loosely coupled.
- Most commercial MIMD computer is loosely coupled.



System Attributes to Performance:-

· The ideal performance of a computer system demands a perfect match between machine capability and program behavior.
· Machine capability can be enhanced with better hardware technology, innovative architectural features and efficient resource management.
· Therefore, program behavior is difficult to predict due to its heavy dependences on application and run time conditions.
· There are also many other factors affecting program behavior, including algorithm design, data structures, language efficiency programmer skill and compiler technology.
· It is impossible to achieve perfect match between hardware and software by merely improving only a few factors without touching other factors.

Some of the factors are as follows:-

1. Clock rate and CPI
2. Performance Factors
3. MIPS Rate
4. Throughput Rate

1. Clock Rate and CPI :-

· The CPU of today's digital computer is driven by a clock with a constant cycle time (τ in nanoseconds).
· The inverse of the cycle time is the clock rate ($f = 1/T$ in megahertz).
· The size of the program is determined by the instruction count (I_C) in terms of the number of machine instructions to be executed in the program.

· Different machine instructions may require different number of clock cycles to execute.
· Therefore, the cycles per instruction (CPI) becomes an important parameter for measuring the time needed to execute each instruction.
· For a given instruction set, we can calculate an average CPI over all instruction types, provided we know their frequencies of appearance in the program.
· An accurate estimate of the average CPI requires a large amount of program code to be traced over a long period of time.
· We simply use the term CPI to mean the average value with respect to a given instruction set and a given program mix.

2. Performance Factors:-

· Let I_C be the number of instructions in a given program or the instruction count.
· The CPU time (τ in seconds per program) needed to execute the program is estimated by finding the product of three contributing factors:-

$$T = I_C * CPI * \tau$$

· The execution of an instruction requires going through a cycle of events involving the instruction fetch, decode, operand(S) fetch, execution and store results.
· A memory cycle is the time needed to complete one memory reference.
· Usually, a memory cycle is k times the processor cycle τ.
· The value of k depends on the speed of the memory technology and processing memory interconnection scheme used.
· The CPI of an instruction type can be divided into two component

terms corresponding to the total processor cycles and needed to complete the execution of the instruction.

· Depending on the instruction type, the complete instruction cycle may involve one of four memory references. Therefore we rewrite as

$$T = I_C * (P + m * k) * \tau$$

Where, P is the number of processor cycles needed for the instruction decode and Execution.

M is the number of memory references needed,

K is the ratio between memory cycle and processor cycle,

I_c is the instruction count,

τ is the processor cycle time.

3. MIPS Rate:-

Let C be the total number of clock cycles needed to execute a given program. Then the CPU time can be estimated as

$T = C * \tau = C/f$

$CPI = C/I_C$ & $T = I_C * CPI$

$T = I_C * CPI/f$

The processor speed is often measured in terms of million instructions per second (MIPS).

4. Throughput Rate:-

· Another important concept related to how many programs a system can execute per unit time, called the system throughput W_s (in programs per second).

· In a multi programmed system, the system throughput is often lower than the CPU throughput W_p defined by,

$W_p = f/(I_c * CPI)$

Note that $W_p = (MIPS) * 10^6 / I_C$. The unit for W_P is programs/second.

THE STATE OF COMPUTING:-

· Early computing was entirely mechanical:
  - abacus (about 500 BC)
  - mechanical adder/subtracter (Pascal, 1642)
  - difference engine design (Babbage, 1827)
  - binary mechanical computer (Zuse, 1941)
  - electromechanical decimal machine (Aiken, 1944)

· Mechanical and electromechanical machines have limited speed and reliability because of the many moving parts. Modern machines use electronics for most information transmission.

Computing Generations:-

· Computing is normally thought of as being divided into generations.

· Each successive generation is marked by sharp changes in hardware and software technologies.

· With some exceptions, most of the advances introduced in one generation are carried through to later generations.

· We are currently in the fifth generation.

First Generation (1945 to 1954):-

· Technology and Architecture:

§ Vacuum tubes and relay memories
§ CPU driven by a program counter (PC) and accumulator
§ Machines had only fixed-point arithmetic

. Software and Applications:

§ Machine and assembly language
§ Single user at a time
§ No subroutine linkage mechanisms
§ Programmed I/O required continuous use of CPU

· Representative systems:
ENIAC, Princeton IAS, IBM 701

Second Generation (1955 to 1964):-

· Technology and Architecture:

§ Discrete transistors and core memories
§ I/O processors, multiplexed memory access
§ Floating-point arithmetic available
§ Register Transfer Language (RTL) developed

· Software and Applications:

§ High-level languages (HLL): FORTRAN, COBOL, ALGOL with compilers and subroutine libraries
§ Still mostly single user at a time, but in batch mode

· Representative systems: CDC 1604, UNIVAC LARC, IBM 7090

Third Generation (1965 to 1974):-

· Technology and Architecture:

§ Integrated circuits (SSI/MSI)
§ Microprogramming
§ Pipelining, cache memories, look ahead processing

· Software and Applications:

§ Multiprogramming and time-sharing operating systems
§ Multi-user applications

· Representative systems: IBM 360/370, CDC 6600, TI ASC, DEC PDP-82

Fourth Generation (1975 to 1990):-

o Technology and Architecture:

- LSI/VLSI circuits, semiconductor memory
- Multiprocessors, vector supercomputers, multi computers
- Shared or distributed memory
- Vector processors

o Software and Applications

- Multi processor operating systems, languages, compilers, and parallel software tools

o Representative systems: VAX 9000, Cray X-MP, IBM 3090, BBN TC2000

Fifth Generation (1990 to present):-

- Technology and Architecture:

o ULSI/VHSIC processors, memory, and switches
o High-density packaging
o Scalable architecture
o Vector processors

- Software and Applications:

o Massively parallel processing
o Grand challenge applications
o Heterogeneous processing

- Representative systems: Fujitsu VPP500, Cray MPP, TMC CM-5,

Intel Paragon

Elements of Modern Computers:-

§ The hardware, software, and programming elements of modern computer systems can be characterized by looking at a variety of factors, including:

- Computing problems
- Algorithms and data structures
- Hardware resources
- Operating systems
- System software support
- Compiler support

Computing Problems:-

Ø Numerical computing:-
- complex mathematical formulations
- tedious integer or floating -point computation

Ø Transaction processing:-
- accurate transactions
- large database management
- information retrieval

Ø Logical Reasoning:-
- logic inferences
- symbolic manipulations

Algorithms and Data Structures:-

§ Traditional algorithms and data structures are designed for sequential machines.

§ New, specialized algorithms and data structures are needed to exploit the capabilities of parallel architectures.

§ These often require interdisciplinary interactions among theoreticians, experimentalists, and programmers.

Hardware Resources:-

Ø The architecture of a system is shaped only partly by the hardware resources.

Ø The operating system and applications also significantly influence the overall architecture.

Ø Not only must the processor and memory architectures be considered, but also the architecture of the device interfaces (which often include their advanced processors).

Operating System:-

Ø Operating systems manage the allocation and deallocation of resources

during user program
execution.

ØUNIX, Mach, and OSF/1 provide support for:-

- multiprocessors and multicomputer
- multithreaded kernel functions
- virtual memory management
- file subsystems
- network communication services

ØAn OS plays a significant role in mapping hardware resources to algorithmic and data structures.

System Software Support:-

ØCompilers, assemblers, and loaders are traditional tools for developing programs in high-level
languages. With the operating system, these tools determine the bind of resources to applications, and the effectiveness of this determines the efficiency of hardware utilization and the system's programmability.

ØMost programmers still employ a sequential mind set, abetted by a lack of popular parallel software support.

System Software Support:-

Ø Parallel software can be developed using entirely new languages designed specifically with parallel support as its goal, or by using extensions to existing sequential languages.

Ø New languages have obvious advantages (like new constructs specifically for parallelism), but require additional programmer education and system software.

Ø The most common approach is to extend an existing language.

Compiler Support:-

Ø Preprocessors:-

- Use existing sequential compilers and specialized
- Libraries to implement parallel constructs

Ø Precompilers:-

- Perform some program flow analysis, dependence
- Checking, and limited parallel optimizations

Ø Parallelizing Compilers:-

- Requires full detection of parallelism in source code, and transformation of sequential code into parallel constructs

Ø Compiler directives are often inserted into source code to aid compiler parallelizing efforts

Evolution of Computer Architecture:-

Ø Architecture has gone through evolutional, rather than revolutional change.

Ø Sustaining features are those that are proven to improve performance.

Ø Starting with the von Neumann architecture (strictly sequential), architectures have evolved to include processing lookahead, parallelism, and pipelining:-

§ Scalar
§ Sequential Lookahead
§ I/E Overlap Functional
§ Parallelism
§ Multiple
§ Func. Units Pipeline
§ Implicit Vector Explicit Vector
§ Memory-tomemory
§ Register-to register
§ SIMD MIMD
§ Associative Processor
§ Array Processor
§ Multicomputer
§ Multiprocessor
§ Architectural Evolution

Flynn's Classification (1972):-

Ø Single instruction, single data stream (SISD): conventional sequential machines
Ø Single instruction, multiple data streams (SIMD): vector computers with scalar and vector hardware
Ø Multiple instructions, multiple data streams (MIMD): parallel computers
Ø Multiple instructions, single data stream (MISD): systolic arrays

Ø Among parallel machines, MIMD is most popular, followed by SIMD, and finally MISD.

Parallel/Vector Computers:-

Ø Intrinsic parallel computers execute in MIMD mode.

Ø Two classes:

- Shared-memory multiprocessors
- Message-passing multicomputer

Ø Processor communication:

- Shared variables in a common memory (multiprocessor)
- Each node in a multicomputer has a processor and a private local memory, and communicates with other processors through message passing.

Pipelined Vector Processors:-

Ø SIMD architecture
Ø A single instruction is applied to a vector (one dimensional array) of operands.
Ø Two families:
Memory -to-memory: operands flow from memory to vector pipelines and back to memory
Register-to -register: vector registers used to interface between memory and functional pipelines.

SIMD Computers:

Ø Provide synchronized vector processing
Ø Utilize spatial parallelism instead of temporal parallelism
Ø Achieved through an array of processing elements (PEs)
Ø Can be implemented using associative memory.

Development Layers (Ni, 1990):-

Ø Hardware configurations differ from machine to machine (even with the same Flynn classification)
Ø Address spaces of processors vary among different architectures, and depend on memory organization, and should match target application domain.
Ø The communication model and language environments should ideally be machine-independent, to allow porting to many computers with minimum conversion costs.
Ø Application developers prefer architectural transparency.

Programming Environments:-

Ø Programmability depends on the programming environment provided to the users.

Ø Conventional computers are used in a sequential programming environment with tools developed for a uni processor computer.

Ø Parallel computers need parallel tools that allow specification or easy detection of parallelism and operating systems that can perform parallel scheduling of concurrent events, shared memory allocation, and shared peripheral and communication links.

Implicit Parallelism:-

Ø Use a conventional language (like C, Fortran, Lisp,or Pascal) to write the program.
Ø Use a parallelizing compiler to translate the source code into parallel code.
Ø The compiler must detect parallelism and assign target machine resources.
Ø Success relies heavily on the quality of the compiler.
Ø Kuck (U. of Illinois) and Kennedy (Rice U.) used this approach.

Explicit Parallelism:-

· Programmer write explicit parallel code using parallel dialects of common languages.
· Compiler has reduced need to detect parallelism, but must still preserve existing parallelism and assign target machine resources.
· Seitz (Cal Tech) and Daly (MIT) used this approach.

Needed Software Tools:-

· Parallel extensions of conventional high-level languages.

· Integrated environments to provide

o different levels of program abstraction
o validation, testing and debugging
o performance prediction and monitoring
o visualization support to aid program development, performance measurement
o graphics display and animation of computational results

CATEGORIES OF PARALLEL COMPUTERS:-

Ø Considering their architecture only, there are two main categories of parallel computers:
§ systems with shared common memories, and
§ systems with unshared distributed memories.

a) Shared-Memory Multiprocessors:-

Ø Shared-memory multiprocessor models:
o Uniform-memory -access (UMA)
o Nonuniform-memory -access (NUMA)
o Cache-only memory architecture (COMA)
Ø These systems differ in how the memory and peripheral resources are shared or distributed.

The UMA Model – 1:

- Physical memory uniformly shared by all processors, with equal access time to all words.
- Processors may have local cache memories.
- Peripherals also shared in some fashion.
- Tightly coupled systems use a common bus, crossbar, or multistage network to connect processors, peripherals, and memories.
- Many manufacturers have multiprocessor (MP) extensions of uniprocessor (UP) product lines.

The UMA Model – 2:

- Synchronization and communication among processors achieved through shared variables in common memory.

- Symmetric MP systems – all processors have access to all peripherals, and any processor can run the OS and I/O device drivers.

- Asymmetric MP systems – not all peripherals accessible by all processors; kernel runs only on selected processors (master); others are called attached processors (AP).