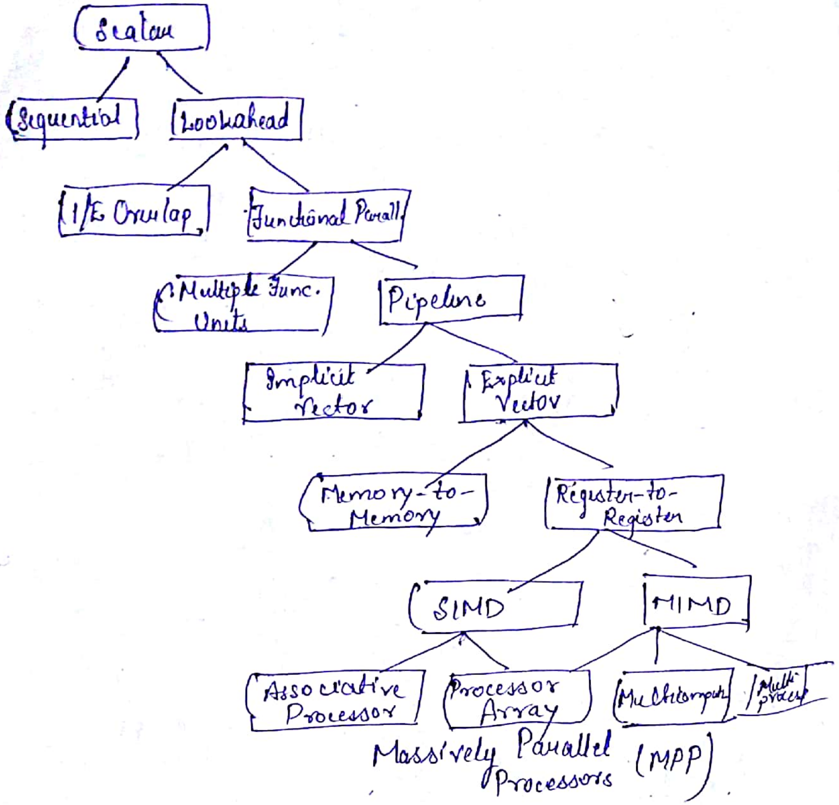
Tree showing architectural evolution



What is vector processors ?
Vector processor is a CPU that implements an instruction set containing instructions that operate on one -D array of data called vectors

Vector

Memory-2-memory Architecture
(Supports the pipelined flow of vector operands directly from memory to pipelines and then back to the memory)

Register-2-register Architecture
(uses vector registors to interface between the memory & functional pipelines)

SIMD uses spatial parallelism rather than temporal parallelism.

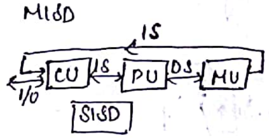
† Progress in hardware :-

1st → vaccum tube
2nd → transistors
3rd → ICs
4th → VLSI
5th → Semiconductors

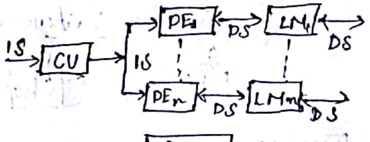
	I_c	p	m	k	T
IS	✓	✓			
CT	✓	✓	✓		
CPU		✓			✓
Cache			✓	✓	✓

† Look-Ahead, Parallelism and Pipelining
prefetch instruction

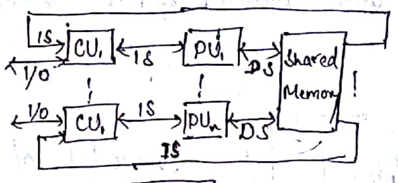
† Flynn's classification :- based on notion of instructions & data streams
SISD (Conventional sequencial machines)
SIMD (Vector computers)
MIMD (Parallel computer)
MISD



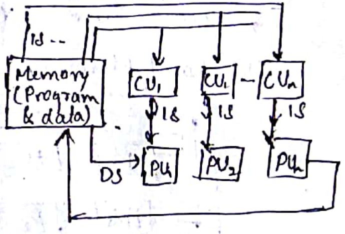
SISD



SIMD



MIMD



MISD

→ Parallel computer are SIMD or MIMD

† System attributes to Performance

$T = I_c \times CPI \times \tau$ (cycle time)

p: decoder & execution.

$T = I_c \times (p + m \times k) \times \tau$

p = no. of processor cycle needed to decode and execute instruction

m = no. of memory reference needed

k = $\frac{\text{memory cycle time}}{\text{processor cycle time}}$

$CPI = C/I_c$

MIPS rate = $\frac{I_c}{T \times 10^6} = \frac{f}{CPI \times 10^6}$

$= \frac{f \times I_c}{C \times 10^6}$

Throughput Rate - $W_p = \frac{f}{I_c \times CPI} = \frac{MIPS \times 10^6}{I_c}$

CPU time = $(I_c \times 10^{-6})/MIPS$

† Implicit and Explicit Parallelism

† Multiprocessors and Multicomputers

† Similarity → Parallel computer,

† Differences:- (Mp)Shared memory, MC (Destrubuted)

Shared Memory multiprocessors / Tightly coupled due to high degree of resource sharing

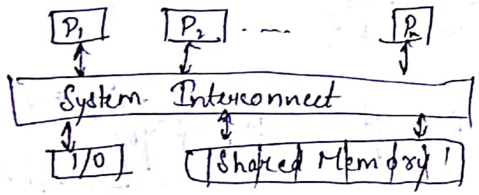
UMA

→ physical memory uniformally distrubuted to all processors.

→ Equal Access time to all memory words

† Symmetric [All processors have equal access to peripherals

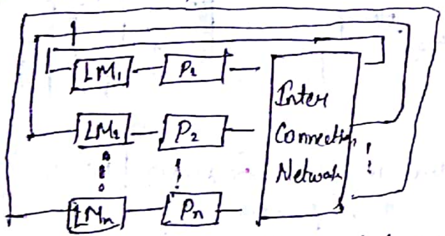
† Asymmetric :- One or subset of processors are executive capable



NUMA

→ access time varies with location of memory word.

LM:- Local Memory



Shared Local Memories

We can also add global access memory

Fastest is Local memory then global, then remote

Cluster-shared memory module:-

Each cluster is UMA or NUMA connected to global shared memory

$T = \frac{2N}{M} + (k+1)\log_2 M$; k-cycles

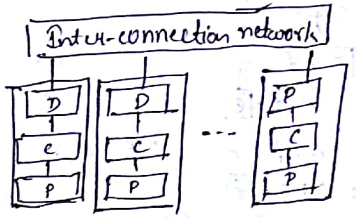
2N- Instruction count

M- processors

COMA:- cache-only memory access

Special case of NUMA in which the distributed main memories are converted to caches.

Remote access is assisted by distrubuted cache directories



P:- processor

C:- cache

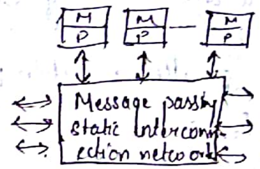
D:- directory

Disadvantage of multiprocessor

→ Scalability

† Distributed memory multicomputers

earlier these multicomputers were called no remote memory access (NORMA)



Multicomputer generation

1st → Hypercube architecture & software controlled message switching

2nd → Mess-Connected architecture, hardware message routing

3rd → fine grain multicomputers with processor and communication gears on same VLSI chip

Multivector and SIMD Computer

Chapter -2.

† Data & Resource dependence

† Data dependence : Ordering relationship between statements

1. Flow (S1 → S2) S2 is flow dependent on S1

$S1 :\rightarrow a+b = \boxed{c}$

$S2 \rightarrow \boxed{c}+d = e$

2. Anti" . (S1 ↛ S2)

$S1 : \boxed{e}+f = g$

$S2 : c+d = \boxed{e}$

3. Output $S1 : a+b = c$ S1 o→ S2

$S2 : d+e = c$

4. I/O dependence : Same file is referenced by by both I/O statement

† Control dependence :- Order of execution of statements can't be determined before run Time.

eg :- if

† Resource dependence :- arises in case of shared memory

† Bernstein's condition for parallel execution :

Processes :- P_1 and P_2

Input :- I_1 and I_2

Output :- O_1 and O_2

$P_1 || P_2$ if $I_1 \cap O_2 = \phi$

$I_2 \cap O_1 = \phi$

$O_1 \cap O_2 = \phi$

Hardware & Software Parallelism :-

HP	SP
1. defined by machine architecture and hardware multiplicity.	defined by the control and data dependence of programs
2. characterizing of parallelism in a processor is by no. of instruction issues per machine cycle.	Flow graph is used to find parallelism checked by data & control parallelism

Solve mismatch between software parallelism and hardware parallelism.

† develop compilation support

† hardware redesign for more efficient exploitation by an intelligent compiler.

Role of compilers :-

→ Remove mismatch between software & hardware parallelism.

→ Increase flexibility in hardware parallelism & to exploit software parallelism. in control intense programs

Program Partitioning & Scheduling

- Grain sizes & latency.

→ Grain size or granularity is measure of the amount of computation involved in a software process. i.e to count the no. of instructions in a grain (program segment).

→ Latency is the time required for communication overhead incurred between machine subsystems eg:-memory latency & synchronization latency