> Parallelism

↳ Exploiting concurrency. Same task divided up to execute concurrently on many processing cores

Parallel vs Distributed computing:

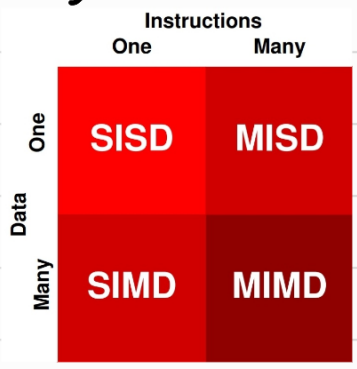
	Parallel	Distributed
No. computers	One	Many
Scalability	Hard to optimise more cores in 1 computer	Easy to add more computers

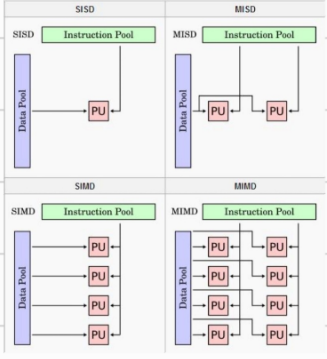
Both can overlap

> Flynn's Taxonomy (naming of Processing Elements (PE's)

Based on relationship b/w PE and:

- Instruction sequence executed
- Memory
- Interconnection network





> SISD

- Serial (non-parallel) computer, traditional von Neumann single CPU PC
  - ↳ One instruction and data stream input / clock cycle
    - Good for sequential tasks
  - ↳ Deterministic Execution (Given same input, always same reaction)

> SIMD

- Type of parallel computer, Vector processors fine grained data
  - ↳ One instruction execution on multiple data elements concurrently.
    - Good for high regularity tasks e.g. Image rendering
  - ↳ Synchronous (clockstep) and deterministic execution
    - ↳ 2 types: Processor arrays and Vector pipelines

one task before next

MISD

↳ Each processing unit operates on data independently via independent instruction streams

↳ Not used often e.g. Multiple algorithms to solve cryptography

MIMD: Most modern and common

↳ Every processor may be executing different instruction stream or data stream

↳ Can be synchronous, asynchronous, non / deterministic

e.g. Multi-core CPU

Memory Architectures

Types: Impacts: - Performance - Reliability - Security

- **Shared Memory**: - Multiple processors access memory via common local bus / switch and address space
  - All processors can access global memory
- **Distributed**: - Processing elements access memory over network/bus/switch, not always common address space
  - Processors all have own local memory, no global space, thus no need for cache coherency → need network
    ↳ Programmer's responsibility to determine exchange of data b/w cores
- **Hybrid**: Combination of above
  - Shared memory component is usually a cache coherent Symmetrical Multi-Processing (SMP) machine.
    ↳ Processors on one SMP access global space.
    ↳ SMPs are independent of each other so still require network

> Parallel Programming Models

- Common models: Shared memory, Threads, Message Passing, Data Parallel and Hybrid
- Models abstract hardware and memory architectures and can be implemented on various hardware types

> Programming Structures

- Single Program Multiple Data
  - ↳ One source program executed independently by multiple processors.
  - ↳ Can be tailored for specific processor roles ex) master-slave

- Multiple Program Multiple Data (Part of MIMD)
  - ↳ Each processor has own program (may contain copies)
  - ↳ Usually involves 2 source programs. One each for master and slave processor

Work Qs

① 1) SISD → For simple sequential tasks
2) SIMD → For repetitive tasks on large data.
↳ Vector pipelines, array processes
3) MISD → Multiple variations on single data.
↳ Cryptography
4) MIMD → Versatile, common for modern CPUs
↳ Multi-purpose.

② 32 execution units → 2 ns each
300 elements needing A - B

$\frac{300}{32} = 9.375 \simeq 10$ batches of executions

$\therefore 10 \times 2 = 20$ ns for all computations

③ - 4 processes, 3 GiB each.
- +1 process every 10 seconds, 0.5 GiB.
- 80 dispatched machines

1) No. processes = 320 all the time.
4+1 process/machine
$5 \times 80 = 400$ processes (F)
2) If 10 are down, processes go to 320
$5 \times 70 = 350$ processes (F)
3) Total memory less than 240 GiBs.
$4 \times 3 + 0.5 = 12.5$ GiB/machine
$\therefore 12.5 \times 80 = 1000$ GiB (F)
4) 20+ machines = less than 1200 GiBs
$12.5 \times 100 = 1250$ GiB (F)