Chapter 5

Mutliprocessors and thread-level parallelism

Flynn's taxonomy of Parallel Systems

Looks at instructions and data parallelism. Oldest (1960's) and best known of many proposals.

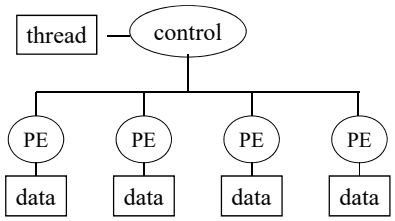
S M	I	S M	D	• S for single • M for multiple
				• I for instruction • D for data.

- SISD is a sequential computer.
- SIMD has one sequence of instructions applied to multiple data.
- MIMD has multiple sequence of instructions executing on multiple data.
- An MISD machine – need to be innovative to define it.

SIMD (two flavors)

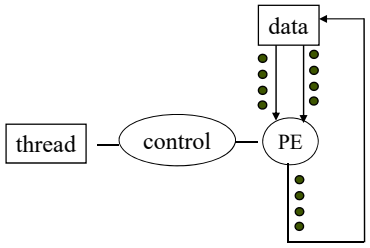
1) Synchronous, lockstep execution

All PEs execute the same instructions on different data

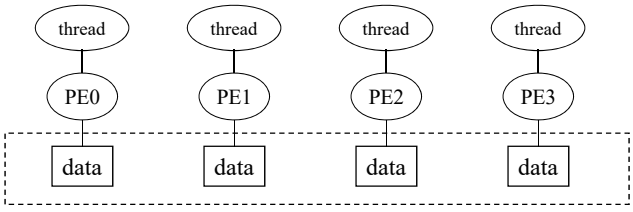


2) Vector processing

The same instruction is repeatedly executed on different data



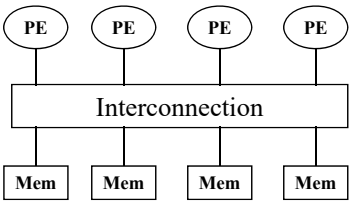
MIMD

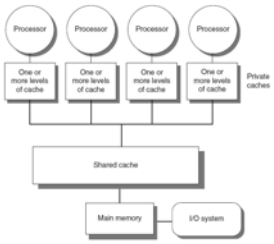


Multiple threads executing on different data – However, if the threads are to cooperate to solve a problem (as opposed to solving different problems), there should be interaction between the programs and/or the data.

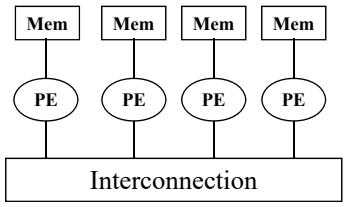
Many flavors depending on the physical **memory architecture** and the virtual **address space** of each processing element (PE).
Address space = the range of memory addresses that the PE can access.

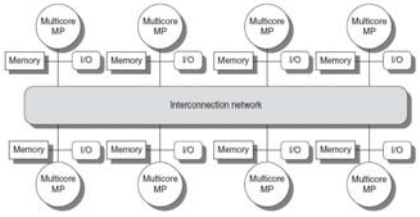
Physical memory Architectures





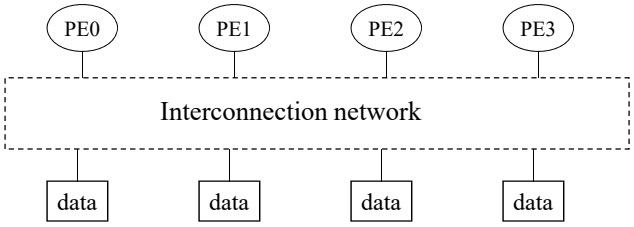
Global, shared memory (Symmetric Multi-Processors – SMP)





Distributed memory

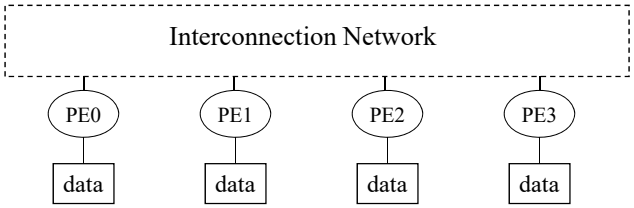
Shared address space



Each of PE0, PE1, PE2 and PE3 can address (directly access) the same locations (shared virtual address space)

- No need for message passing – communicate through shared memory locations.

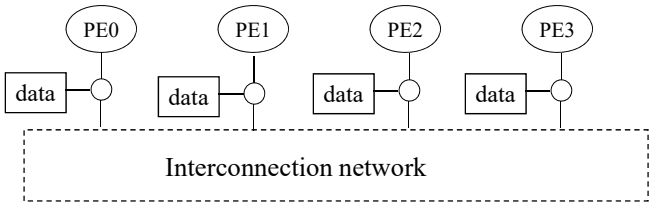
Distributed address space



Each of PE0, PE1, PE2 and PE3 have separate virtual address spaces.

- In order for PEi to access data in the address space of PEj, the two processes have to communicate through explicit messages.

Distributed shared memory systems



Shared address space, but physically distributed memory.

- No need for message passing – communicate through shared memory locations.
- Data is physically distributed, but a runtime system is responsible to access data that do not reside in the local memory.

Results in the so called "Non Uniform Memory Access" – NUMA
(as opposed to UMA, "Uniform Memory Access")