COMPUTER ARCHITECTURE

Code: CT173

Part VII: Advanced Parallelism Computers

MSc. NGUYEN Huu Van LONG

Department of Computer Networking and Communication,

College of Information & Communication Technology,

CanTho University

Agenda

- **Multithreaded Processor**
  - Fine grained Multithreading
  - Coarse grained Multithreading
  - Simultaneous Multithreading
- **Multicore Processor**
- Flynn Taxonomy for Parallel Processor
- **Single Instruction Multiple Data SIMD**
  - Vector Processor
  - General Purpose Graphical Processing Unit GP-GPUs
- **Multiprocessor**
  - Symmetric vs Asymmetric Multiprocessing
  - Homogeneous vs Heterogeneous Multiprocessor
  - Tightly Multiprocessor
    - Uniform Memory Access UMA
    - Non uniform Memory Access - NUMA
    - Cache Only Memory Access - COMA
  - Loosely Multiprocessor
    - Cluster or Multicomputer

Multithreaded processor: Thread Level Parallelism

- The main limitation of ILP is degree of intrinsic parallelism in the instruction stream --> we could use a simpler core (compared to Superscalar) to exploit Thread Level Parallelism → **Multithreaded processor**, i.e. Sun Niagara T1
- Main advantage: when a thread is stall by some reasons, the other threads can use the idle computing resources → faster overall execution
  - If thread 1 is blocked, the CPU still has a chance of running thread 2 in order to keep the hardware fully occupied
- Multithreaded processor could execute multiple process or thread concurrently, appropriately supported by the operating system
- Multithreaded processor should switch between the threads → the state of each thread could be preserved → need multiple Register Files and multiple PCs
- Multiple techniques are presented in multithreaded processor:
  - **Fine-grained multithreading:**
  - **Coarse-grained multithreading**
  - **Simultaneous multithreading (Upgraded coarse grained multithreading)**

Multithreaded Processor Classification

- **Coarse-grained Multithreading**: when a thread is stalled, perhaps for a cache miss, another thread can be executed;
- **Fine-grained Multithreading**: switching from one thread to another thread on each instruction;
- **Simultaneous Multithreading**: multiple thread are using the multiple issue slots in a single clock cycle.
- The Sun T1 and T2 (aka Niagara) processors are fine-grained multithreaded processors, while the Intel Core i7 and IBM Power7 processors use SMT

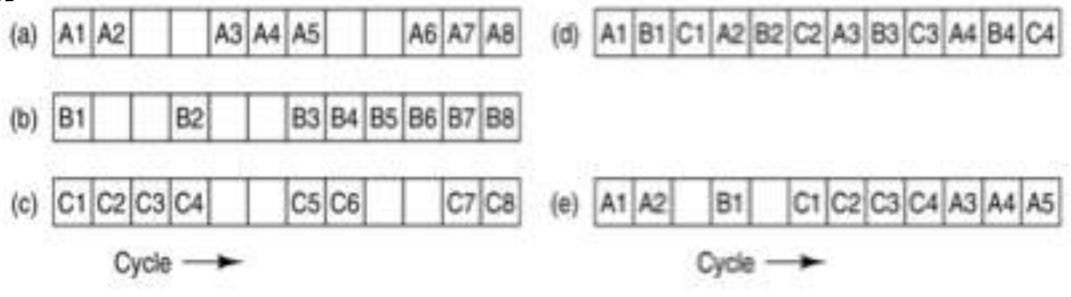

(a) A1 A2 A3 A4 A5 A6 A7 A8
(b) B1 B2 B3 B4 B5 B6 B7 B8
(c) C1 C2 C3 C4 C5 C6 C7 C8
(d) A1 B1 C1 A2 B2 C2 A3 B3 C3 A4 B4 C4
(e) A1 A2 B1 C1 C2 C3 C4 A3 A4 A5
Cycle
Cycle

Figure 8-7. (a)–(c) Three threads. The empty boxes indicate that the thread has stalled waiting for memory. (d) Fine-grained multithreading. (e) Coarse-grained multithreading.

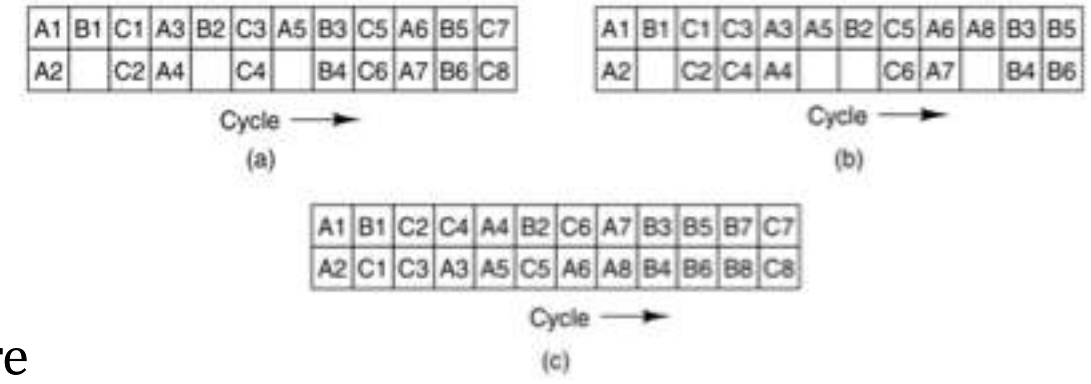
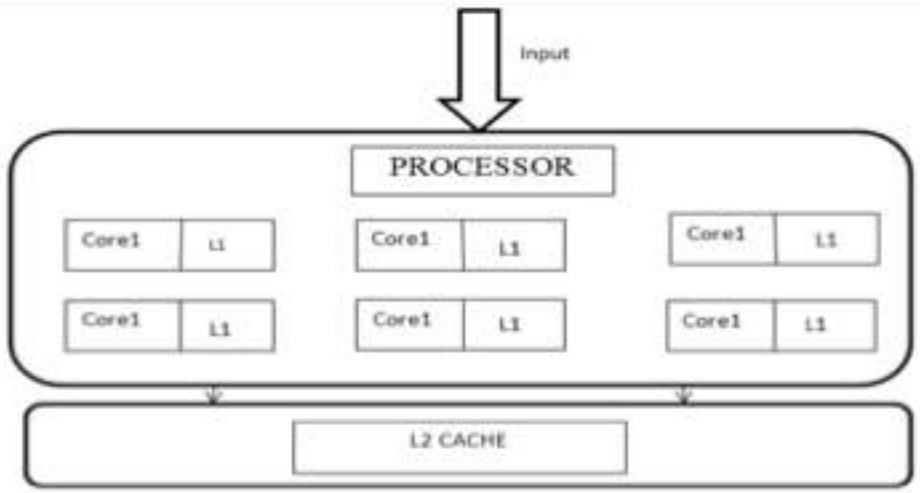

A1 B1 C1 A3 B2 C3 A5 B3 C5 A6 B5 C7
A2 C2 A4 C4 B4 C6 A7 B6 C8
Cycle
(a)
A1 B1 C1 C3 A3 A5 B2 C5 A6 A8 B3 B5
A2 C2 C4 A4 C6 A7 B4 B6
Cycle
(b)
A1 B1 C2 C4 A4 B2 C6 A7 B3 B5 B7 C7
A2 C1 C3 A3 A5 C5 A6 A8 B4 B6 B8 C8
Cycle
(c)

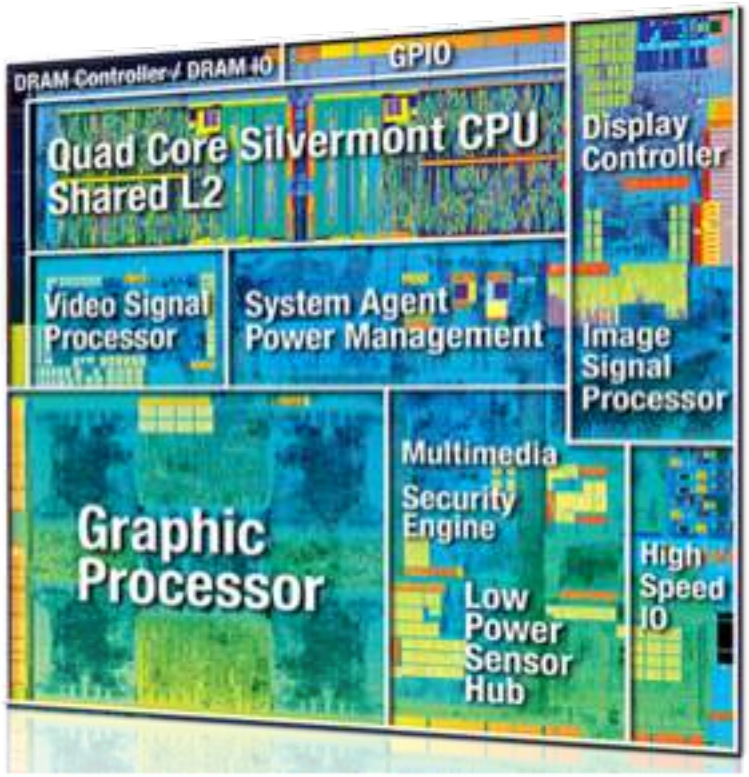
Figure 8-8. Multithreading with a dual-issue superscalar CPU. (a) Fine-grained multithreading. (b) Coarse-grained multithreading. (c) Simultaneous multithreading.

Multicore Processor

- A **Multicore processor** is a single computing component with two or more independent multithreaded processing units (called cores) could read and execute program instructions at the same time
- The cores are integrated onto a single integrated circuit die.
- The **Microprocessors** currently used in almost all PCs are Multicore
- Multi-core processors are widely used across many application domains, including general purpose, embedded, network, digital signal processing (DSP), and graphic (GPU)
- Multicore processors could be a set of homogeneous or heterogeneous cores, e.g. big.LITTLE for heterogeneous and Intel Atom Z3580 for homogeneous
- Cores could be grouped in a Multicore device tightly or loosely



Homogeneous and Heterogeneous Multicore Processor



64 bit Intel Atom Z3580 chip uses homogeneous Multicore processor which combines:

- 4 identical cores that have same performance, power consumption

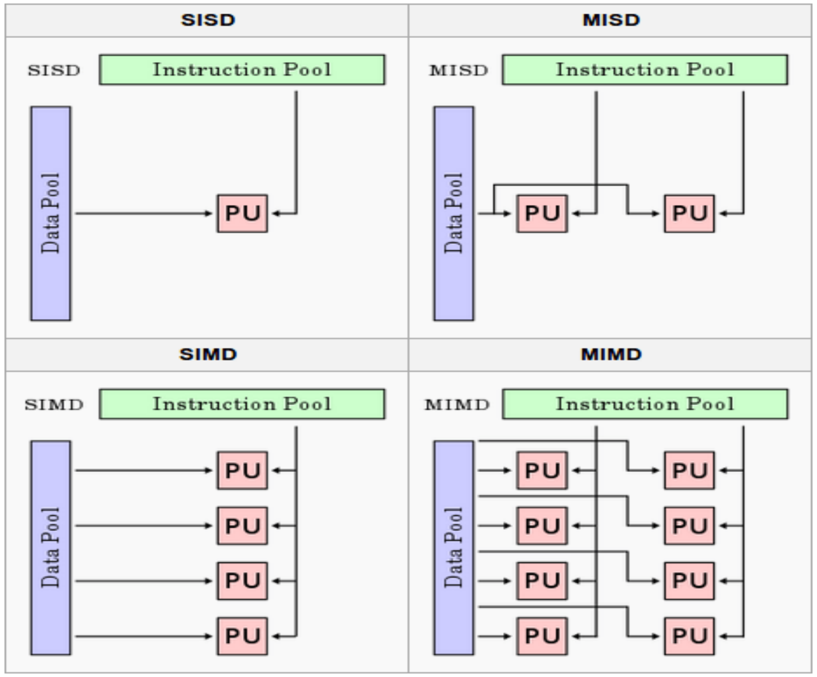
Cortex A57/A53 MPCore big.LITTLE CPU chip uses ARM big.LITTLE, a heterogeneous multicore processor which combines:

- Battery saving and slower cores A53 (LITTLE)
- Powerful and power hungry cores A57 (Big)

Use cases: Snapdragon 810, Exynos 7420

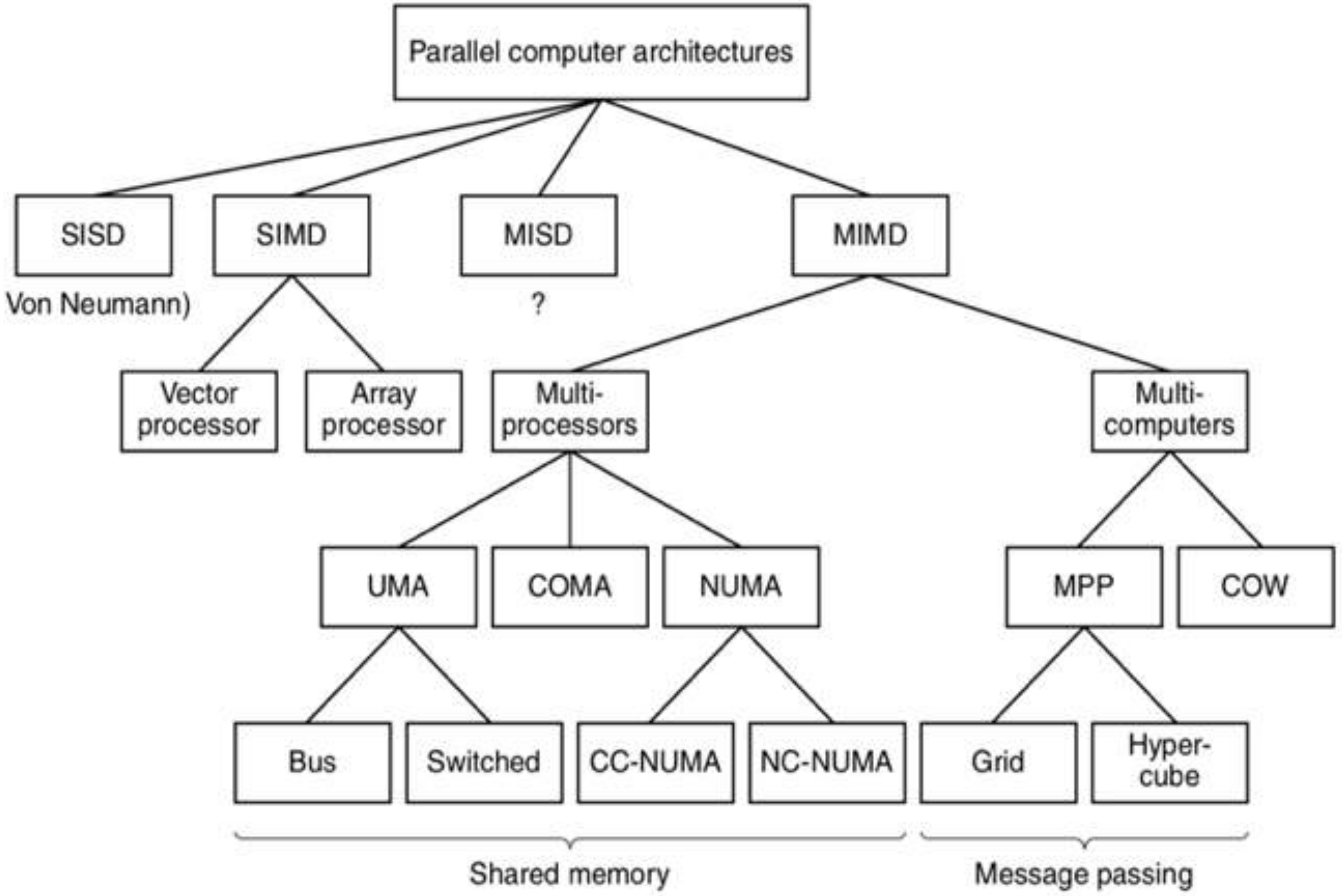
Taxonomy of Parallel Processor

- **Flynn's taxonomy** is a classification of computer architecture, proposed by Michael J.Flynn in 1966. Since the rise of multiprocessing CPUs,. a multiprogramming context has evolved as an extension of the classification system
- The four classifications defined by Flynn are based upon the number of concurrent instruction (or control) streams and data streams available in the architecture



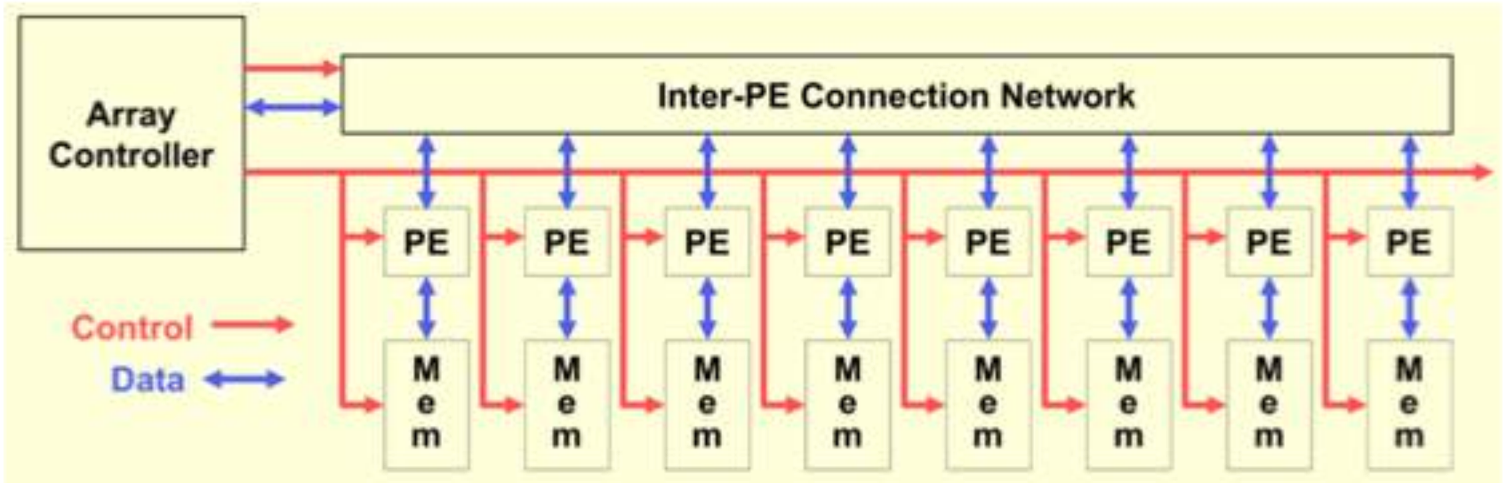
Instruction streams	Data stream	Name	Examples
1	1	**SISD** - Single Instruction Single Data	Uniprocessor, ILP processor (Pipeline, SuperScalar, VLIW)
1	Multiple	**SIMD** - Single Instruction Multiple Data	Vector processor, Multimedia extensions, Graphic processor
Multiple	1	**MISD** - Multiple Instruction Single Data	No commercial implementation
Multiple	Multiple	**MIMD** - Multiple Instruction Multiple Data	Multiprocessor, Multicomputer

Flynn's taxonomy hierarchy



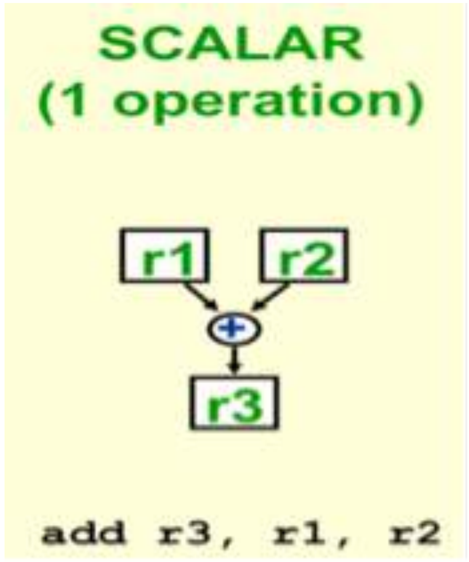
Single Instruction Multiple Data SIMD

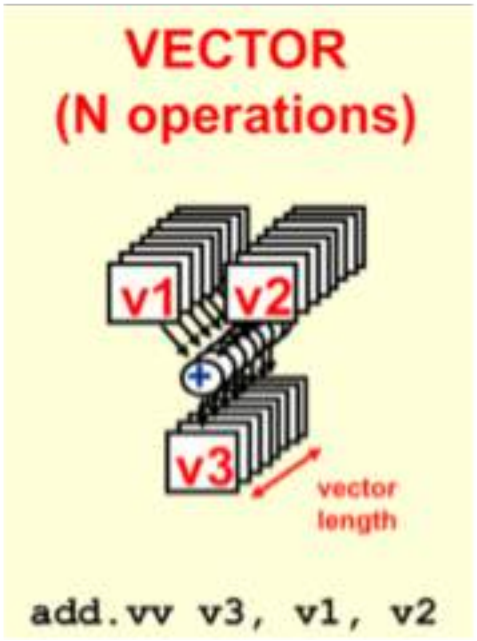
- SIMD architectures can exploit significant data-level parallelism for:
  - Matrix oriented scientific computing
  - Media oriented image and sound processors
- SIMD is more energy efficient than MIMD
  - Only needs to fetch one instruction per data operation
  - Makes SIMD attractive for personal mobile devices
- SIMD allows programmer to continue to think sequentially (compared to MIMD) and achieve parallel speedups
- Three variations: Vector processor, SIMD extension (MMX 1996, SSE, AVX), GPU



Vector processor

- **Vector processor** or **array processor** implements an instruction set that operates on one dimensional arrays of data called *vectors*
- Basic idea:
  - Load **sets** of data elements into ***"vector registers"***
  - Operate on those registers
  - Disperse the results back into memory
- A single instruction operates on ***vectors of data***
  - Synchronized units: ***single Program Counter***
  - Which results in dozens of register-to-register operations
  - Used to hide memory latency (memory latency occurs one per vector load/store vs. one per element load/store)
  - Leverage memory bandwidth



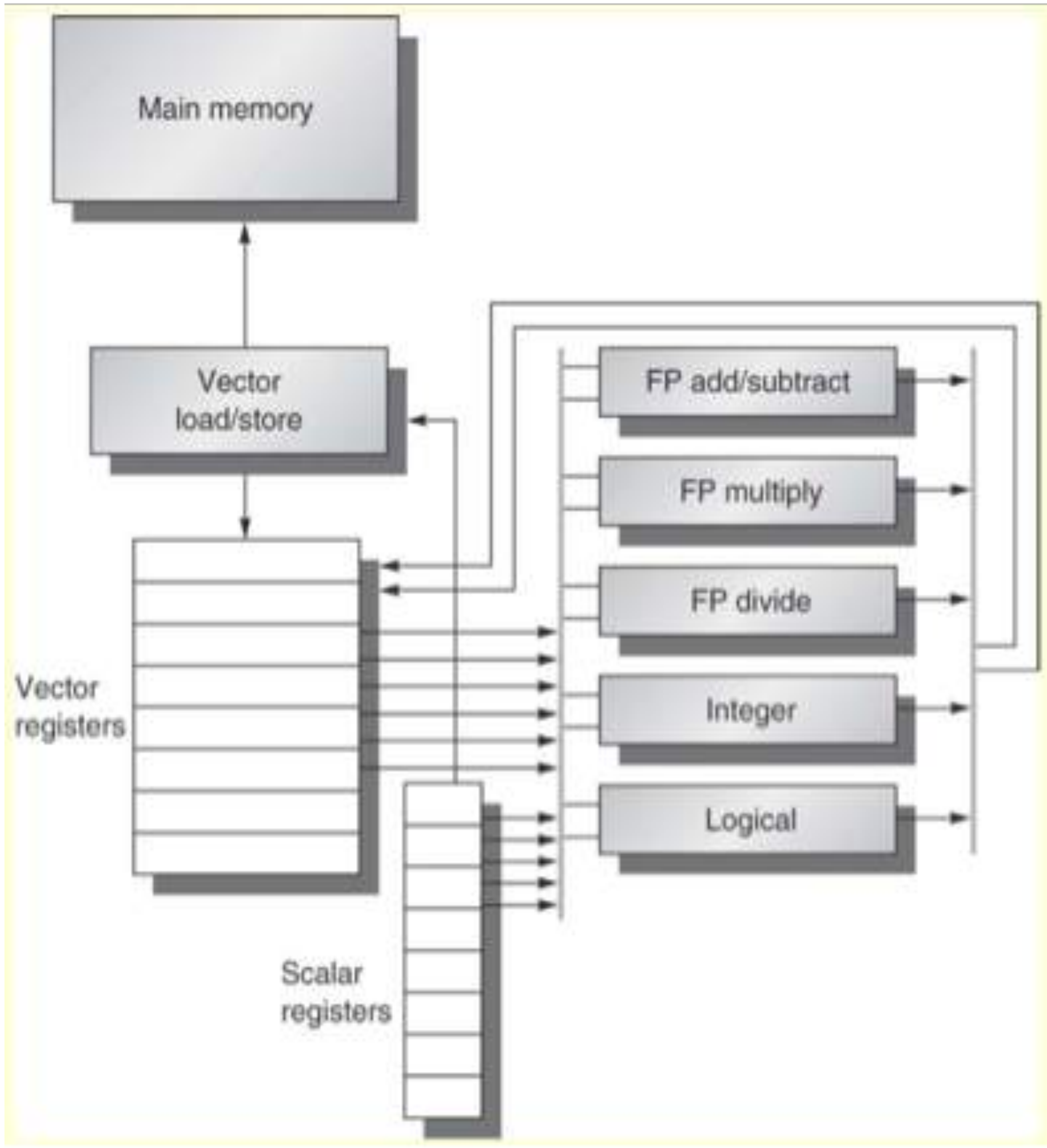


Vector processor properties

- Each result independent of previous result
  - Long pipeline, compiler ensures no dependencies
  - High clock rate
- Vector instructions access memory with known pattern
  - Highly interleaved memory
  - Amortize memory latency of over 64 elements
  - No (data) caches required! (Do use instruction cache)
- Reduces branches and branch problems in pipelines
- Single vector instruction implies lots of work (loop)
  - Fewer instruction fetches
- There are two styles:
  - ***Memory - memory vector processors:*** all vector operations are memory to memory
  - ***Vector - register processors:*** all vector operations between vector registers (except load and store)

Vector processor: VMIPS

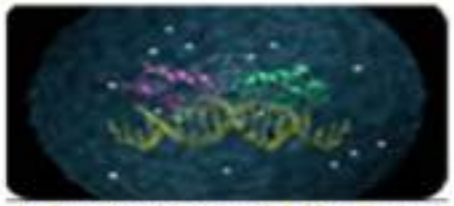
- Loosely based on Cray-1
- *Vector registers*
  - 8 registers. Each register holds a 64-element, 64 bits/element vector
  - Register file has (at least) 16 read ports and 8 write ports
- *Vector functional units*
  - Fully pipelined so they can start a new operation every cycle
- *Vector load-store unit*
  - Fully pipelined, one word per clock cycle after initial memory latency
- *Scalar registers*
  - 32 general-purpose registers
  - 32 floating-point registers



General purpose GPUs (GP-GPUs)

- In 2006, NVIDIA introduced GeForce 8800 GPU supporting a new programming language:
  - ***CUDA,*** "Compute Unified Device Architecture"
  - Subsequently, broader industry pushing for ***OpenCL***, a vendor -neutral version of same ideas for multiple platforms
- ***Basic idea:*** Take advantage of GPU computational performance and memory bandwidth to accelerate some kernels for general-purpose computing.
- Attached processor model: Host CPU issues data-parallel kernels to GP-GPU device for execution
- ***Basic concepts:***
  - *Heterogeneous* execution model: CPU is the ***host***, GPU is the ***device***
  - Develop a C-like programming language for GPUs
  - Unify all forms of GPU parallelism as *CUDA thread*
  - Programming model is *"Single Instruction Multiple Thread" (SIMT)*: massive number of ***light-weight*** threads
  - Programmer unware of number of parallel cores

GP-GPUs could do


Bio-Informatics and Life Sciences

Computational Electromagnetics and Electrodynamics

Computational Finance

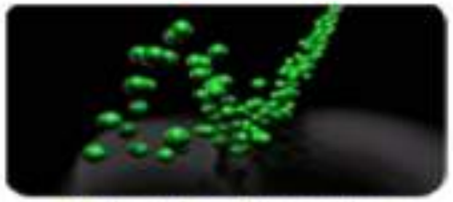
Computational Fluid Dynamics

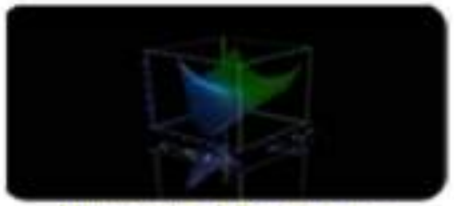
Data Mining, Analytics, and Databases

Imaging and Computer Vision

MATLAB Acceleration

Medical Imaging

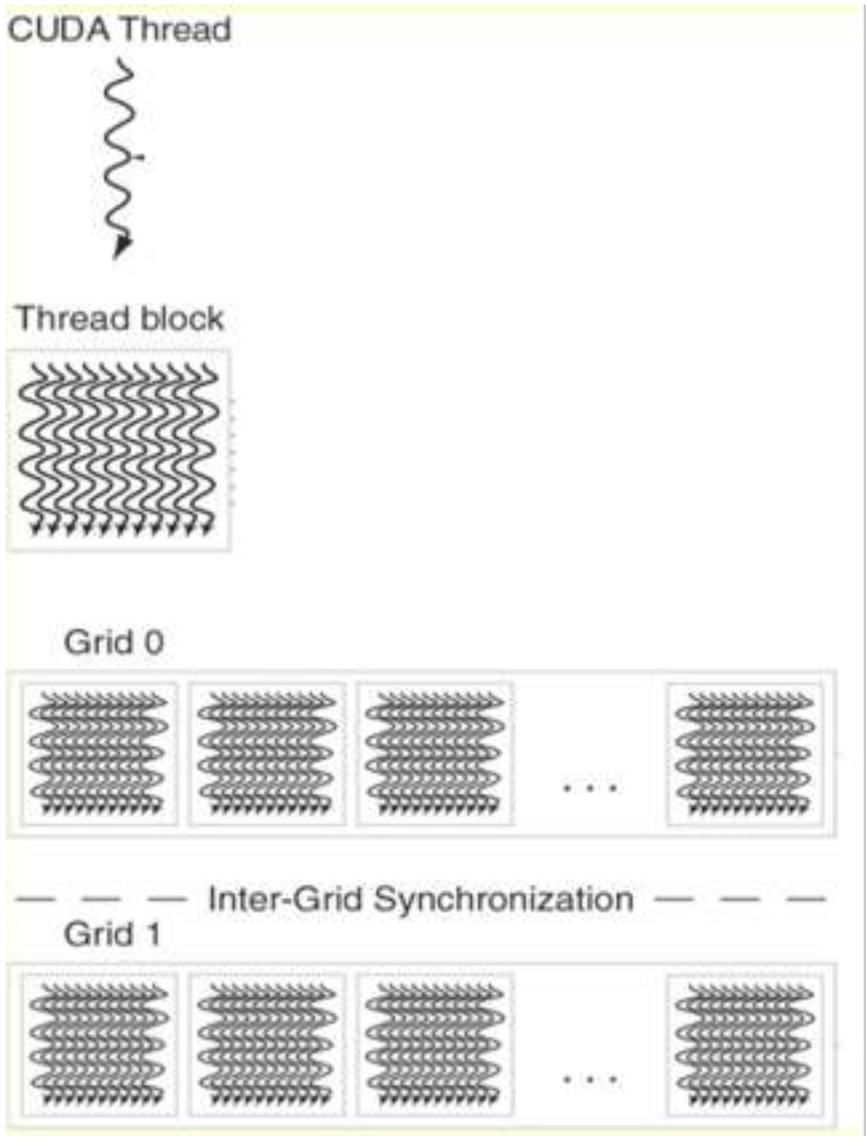

Molecular Dynamics


Numerical Packages

Weather, Atmospheric, Ocean Modeling, and Space Sciences

GP-GPU operation

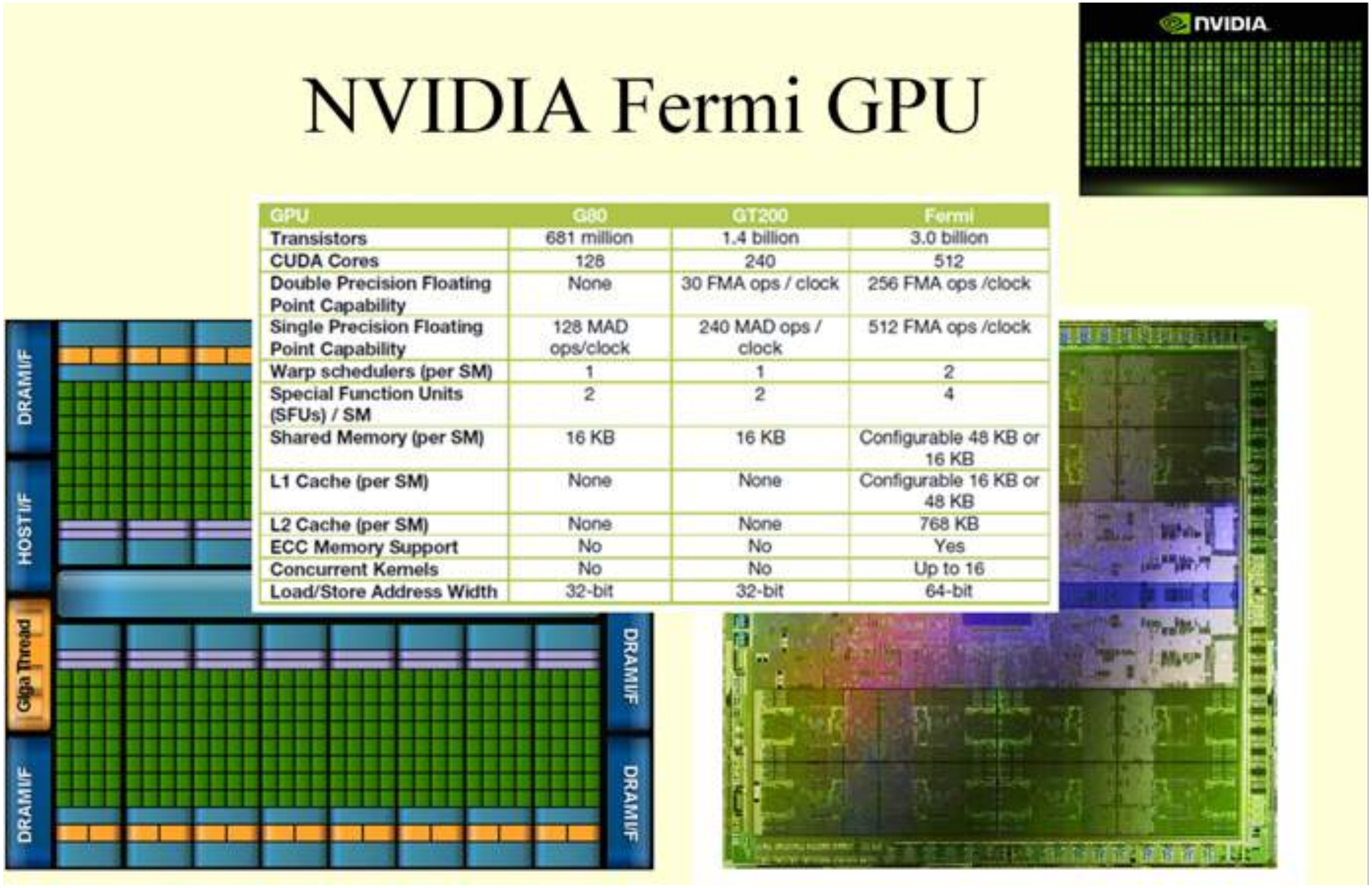
- **Thread**: lower level of parallelism as programming primitive
  - Each thread is composed of SIMD instructions
  - A thread is associated with each data element
- Threads are organized into **Blocks**. Blocks are organized into **Grids**
- Each *thread of SIMD instructions* calculates 32 elements for each instruction.
- Each thread block is executed by a multi-threaded SIMD processor.
- Example: Multiply two vectors of length 8192
  - Each SIMD instruction executes 32 elements at a time
  - Each thread block contains 16 threads of SIMD instructions
  - Grid size = 16 thread blocks
  - Total 16 x 16 x 32 = 8192 elements



GP-GPU vs Vector Processor

- **Similarities**
  - Works well with data-level parallel problems
  - Scatter-gather transfers
  - Mask registers
  - Branch hardware uses internal masks
  - Large register files
- **Differences**
  - No scalar processor
  - Uses multithreading to hide memory latency
  - Has many functional units, as opposed to a few deeply pipelined units like a vector processor

A model of GP-GPU from NVidia

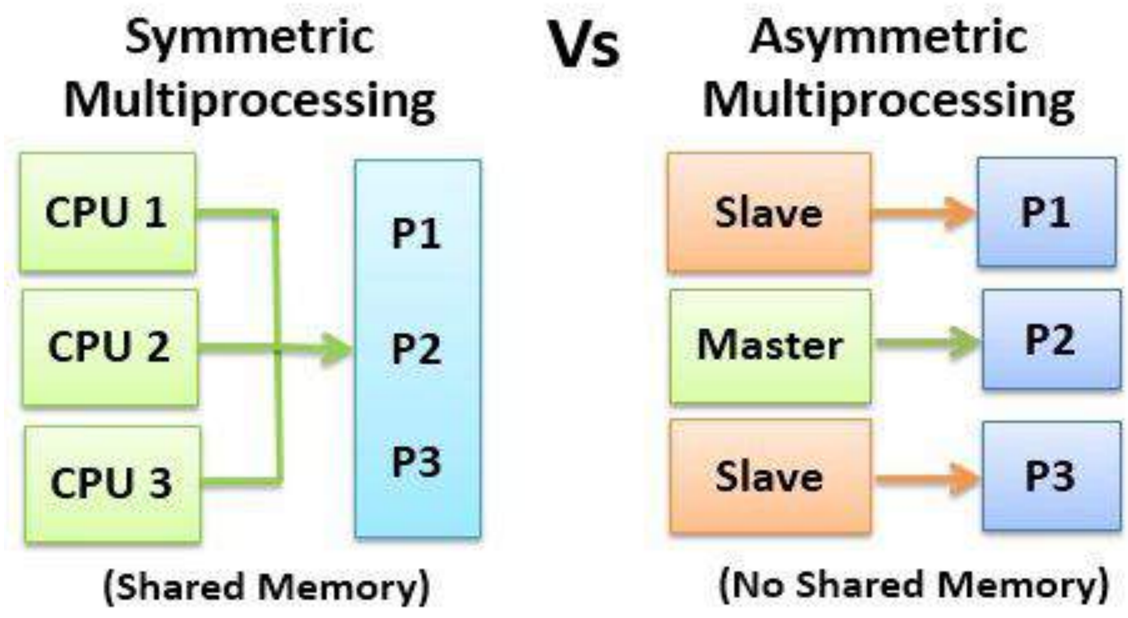

NVIDIA
DRAM I/F
HOST I/F
Giga Thread
DRAM I/F
DRAM I/F
DRAM I/F

NVIDIA Fermi GPU

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Warp schedulers (per SM)	1	1	2
Special Function Units (SFUs) / SM	2	2	4
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache (per SM)	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

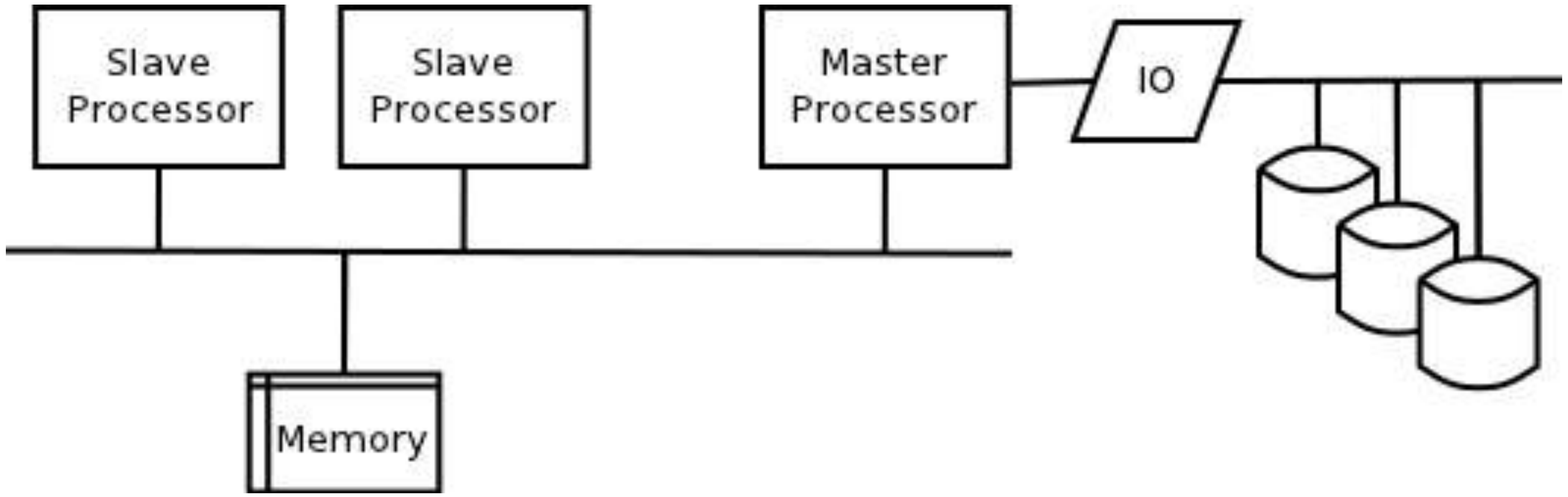
Multiprocessing

- **Multiprocessing** is the use of two or more CPUs (two or more processors), each sharing main memory through a shared address space and peripherals in order to simultaneously process programs, within a single computer system, i.e. a died silicon chip.
- Multiprocessing is classified as MIMD by using Flynn Taxonomy
- There are two main variation techniques for Multiprocessing:
  - Symmetric multiprocessing SMP
  - Asymmetric multiprocessing AMP



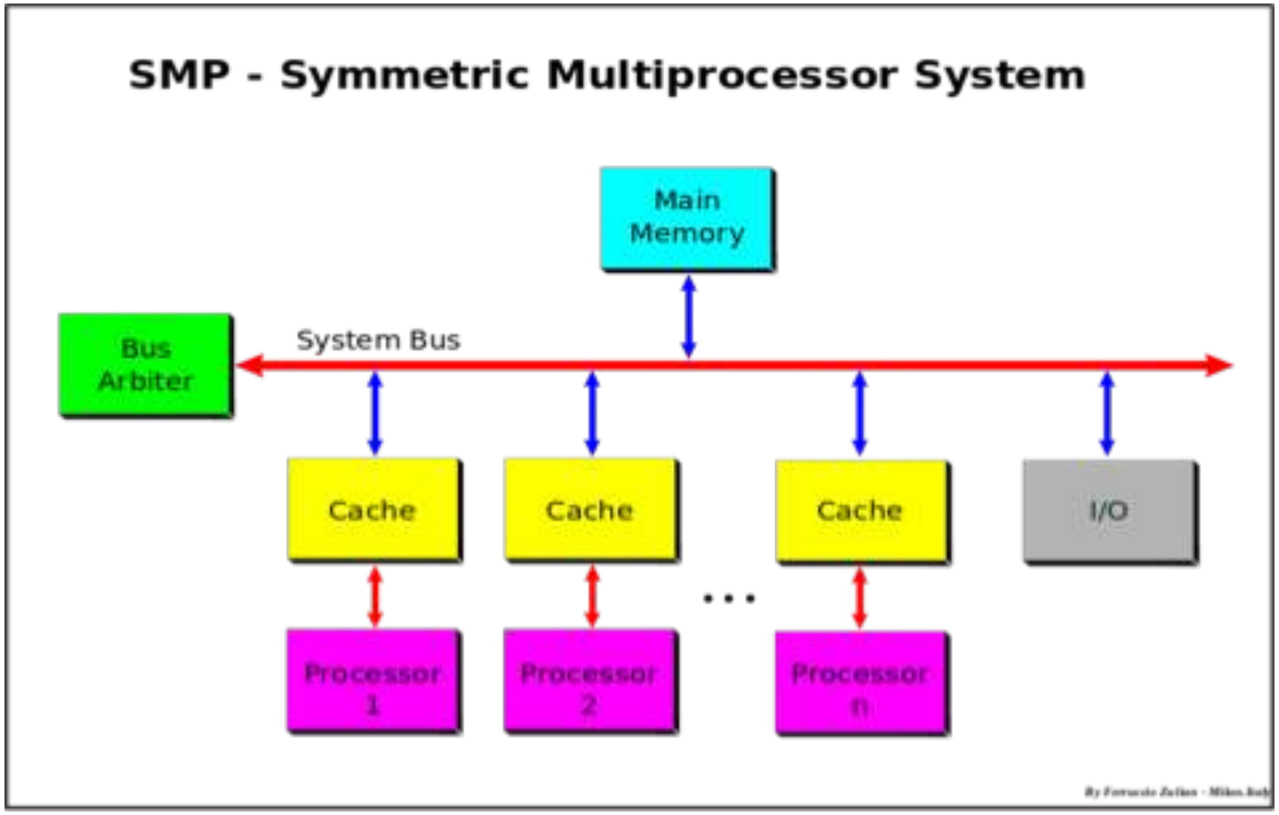
Asymmetric Multiprocessing

- **Asymmetric Multiprocessing (AMP)** is the the only method for handling multiple CPUs before Symmetric Multiprocessing SMP was available
  - Master - Slave relationship among the processors
  - Two or more processors have different access levels, e.g. only one CPU execute operating system code and only one CPU performs I/O operations
  - A method to connect the processors
  - Used in dedicated applications, embedded systems
  - There is no shared memory because processors need not to communicate



Symmetric Multiprocessing

- **Symmetric multiprocessing (SMP)** is the the most used architecture in Multiprocessor today
  - Two or more processors have full access to all input and output devices equally
  - A method to connect the processors
  - A shared memory where all processors have uniform memory access cost
  - Operating system treats all processors equally

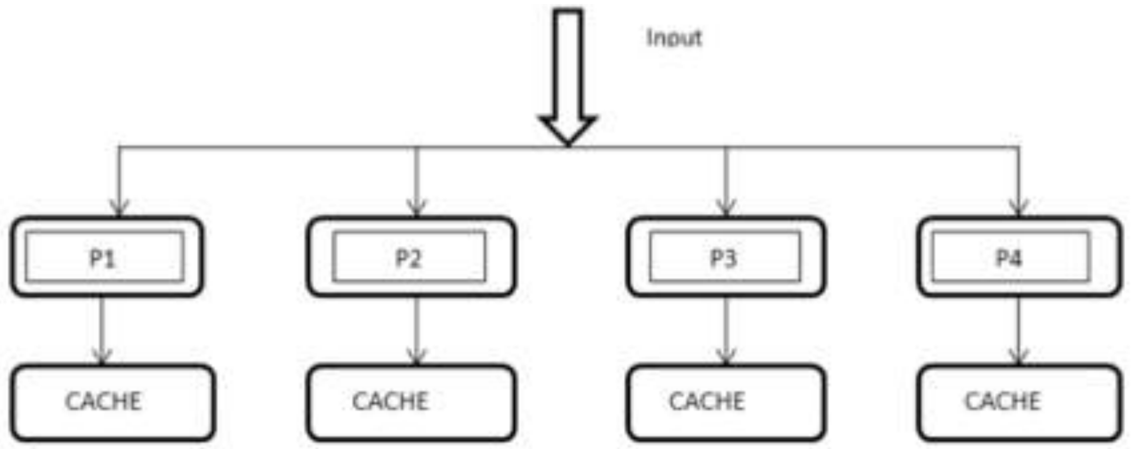


Symmetric vs Asymmetric Multiprocessing

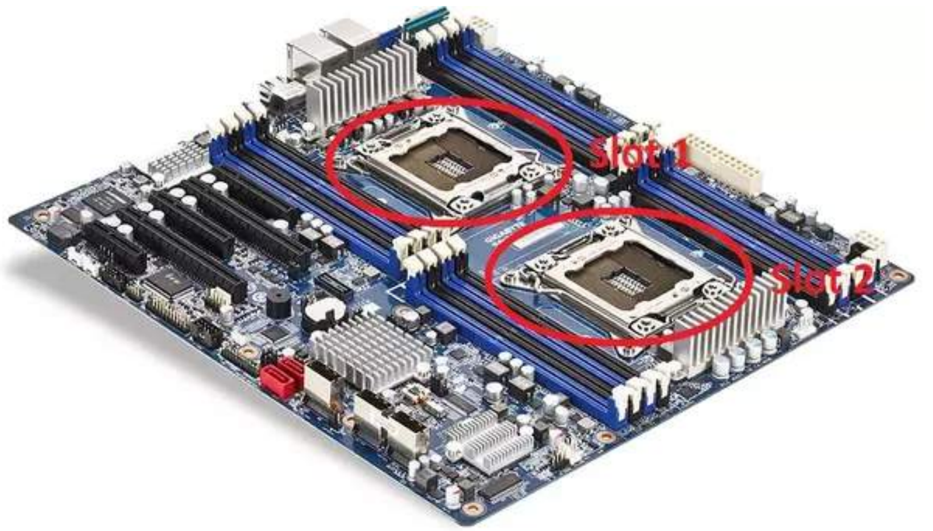
BASIS FOR COMPARISON	SYMMETRIC MULTIPROCESSING	ASYMMETRIC MULTIPROCESSING
Basic	Each processor run the tasks in OS	Only Master processor run the tasks of OS
Process	Processor takes processes from a common ready queue, or there may be a private ready queue for each processor	Master processor assign processes to the slave processors or they have some predefined processes
Architecture	All processor in SMP has the same architecture	All processor in AMP may have same or different architecture
Communication	All processors communicate with another processor by a shared memory	Processors need not communicate as they are controlled by the master processor
Failure	If a processor fails, the computing capacity of the system reduces	If a master processor fails, a slave is turned to the master processor to continue the execution. If a slave processor fails, its task is switched to other processors
Ease	Symmetric Multiprocessor is complex as all the processors need to be synchronized to maintain the load balance	Asymmetric Multiprocessor is simple as master processor access the data structure

Multiprocessor: Homogeneous and Heterogeneous

- Multiprocessor is a system which could use two complementary techniques as a combination: **Multithreading** and **Multiprocessing**
- Multiprocessor now is a play a major role from embedded to high end general-purpose computing: very high-end performance, scalability, reliability
- Multiprocessor could be a set of homogeneous or heterogeneous processors
  - **Homogeneous**: the processors are identical and perform exactly the same tasks and have exactly the same capabilities available
  - **Heterogeneous**: the processors are a mixture of design and could be differ in capabilities, speed and perform a task differently
- Multiprocessor used mainly in high performance servers to complete heavy tasks, i.e. Supercomputer Cray XD1



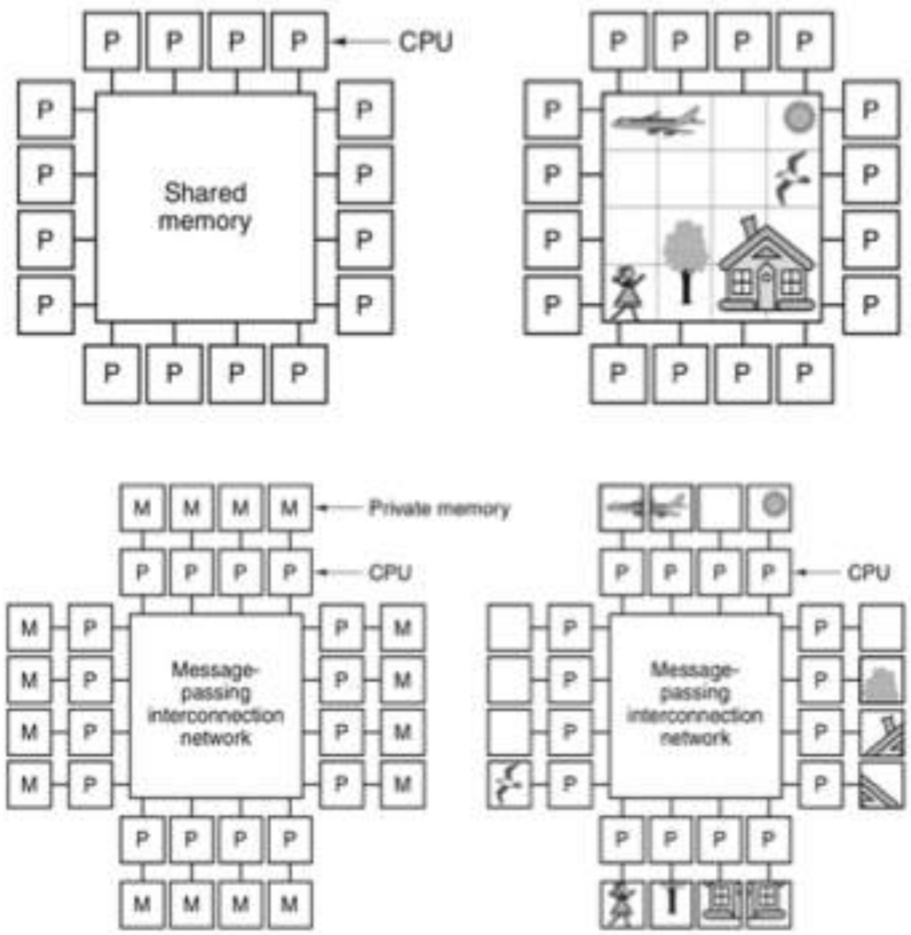
Multiprocessor: A view of technology



High end workstation using a board which supports 2 or 4 or more slot for the CPUs. These CPUs should work parallel in order to complete the complex tasks. Intel Xeon, AMD Opteron are the good example for Multiprocessor

Multiprocessor: Tightly and Loosely

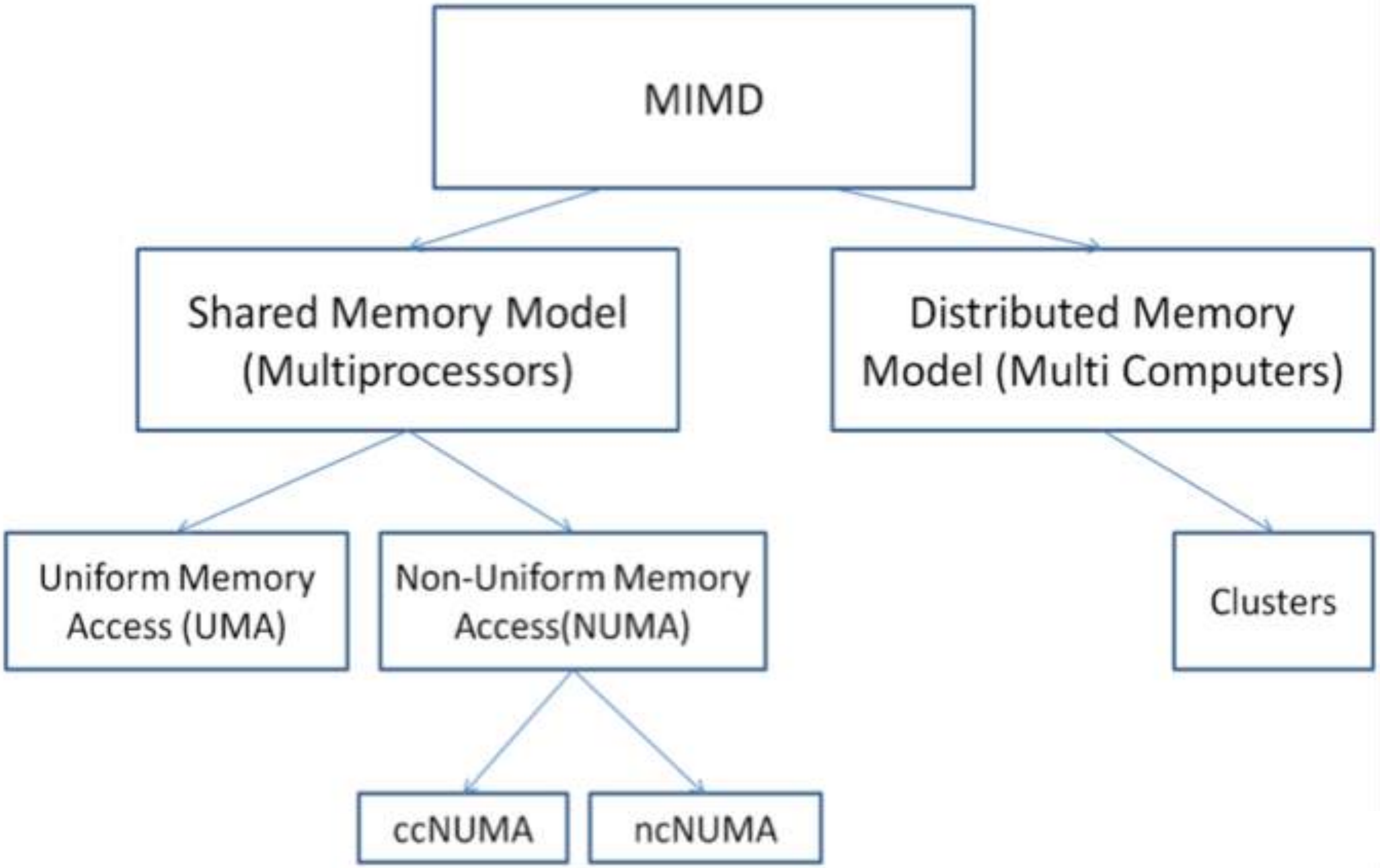
- Multiprocessor could be done in two architectures
  - Tightly coupled Multiprocessor system in which multiple processors that are connected at the bus level. A memory is shared to access among CPUs
  - Loosely coupled Multiprocessor system or called Clusters in which multiple standalone single or dual processor interconnected via a high speed communication system (Gigabit Ethernet is common). Each processor manage it own local memory (distributed memory)
- Tightly coupled multiprocessor system is introduced by
  - Uniform Memory Access - UMA
  - Non uniform Memory Access - NUMA
    - Non cache Uniform Memory Access NC-NUMA
    - Cache Coherent Uniform Memory Access CC-NUMA
  - Cache Only Memory Access - COMA


CPU
Shared memory
Private memory
CPU
Message-passing interconnection network
CPU
Message-passing interconnection network

Multiprocessor vs Multicore Processors

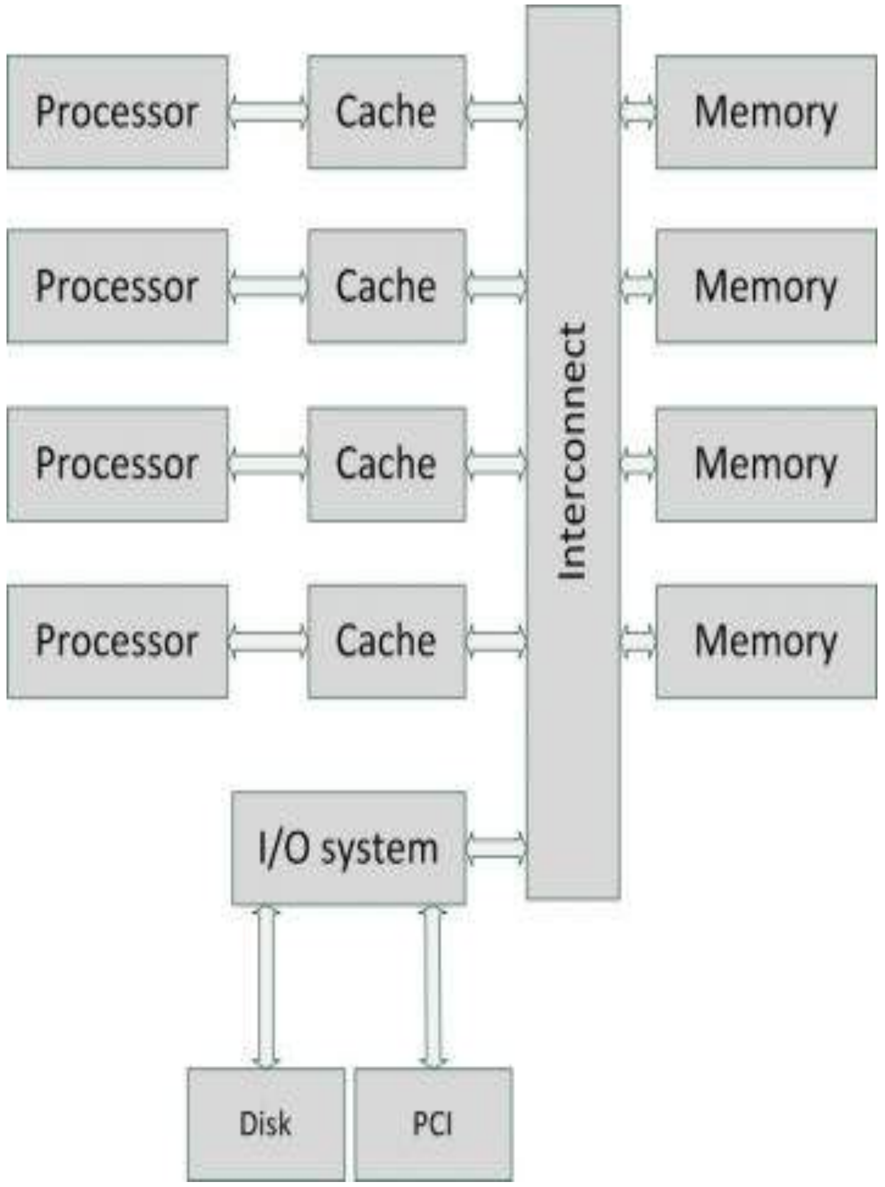
Multicore	Multiprocessor
Single or multiple integrated circuit die/s	Single or multiple systems
Cheaper (single CPU that does not require multiple CPU support system)	Expensive (Multiple separate CPU's that require a system that supports Multiprocessor)
Will have less traffic (cores integrated into a single chip and will require less time)	Will have more traffic (distances between the two will require a longer time)
Does not need to be configured	Needs a little complex configuration
Faster running a single program	Faster running multiple programs
Only one chip to reduce the energy used to send signals from chip to chip	Spend more energy in associated wiring between chips on a circuit board
Save circuit-board and packaging space	

Multiprocessor: Flynn's classification


MIMD
Shared Memory Model (Multiprocessors)
Distributed Memory Model (Multi Computers)
Uniform Memory Access (UMA)
Non-Uniform Memory Access(NUMA)
Clusters
ccNUMA
ncNUMA

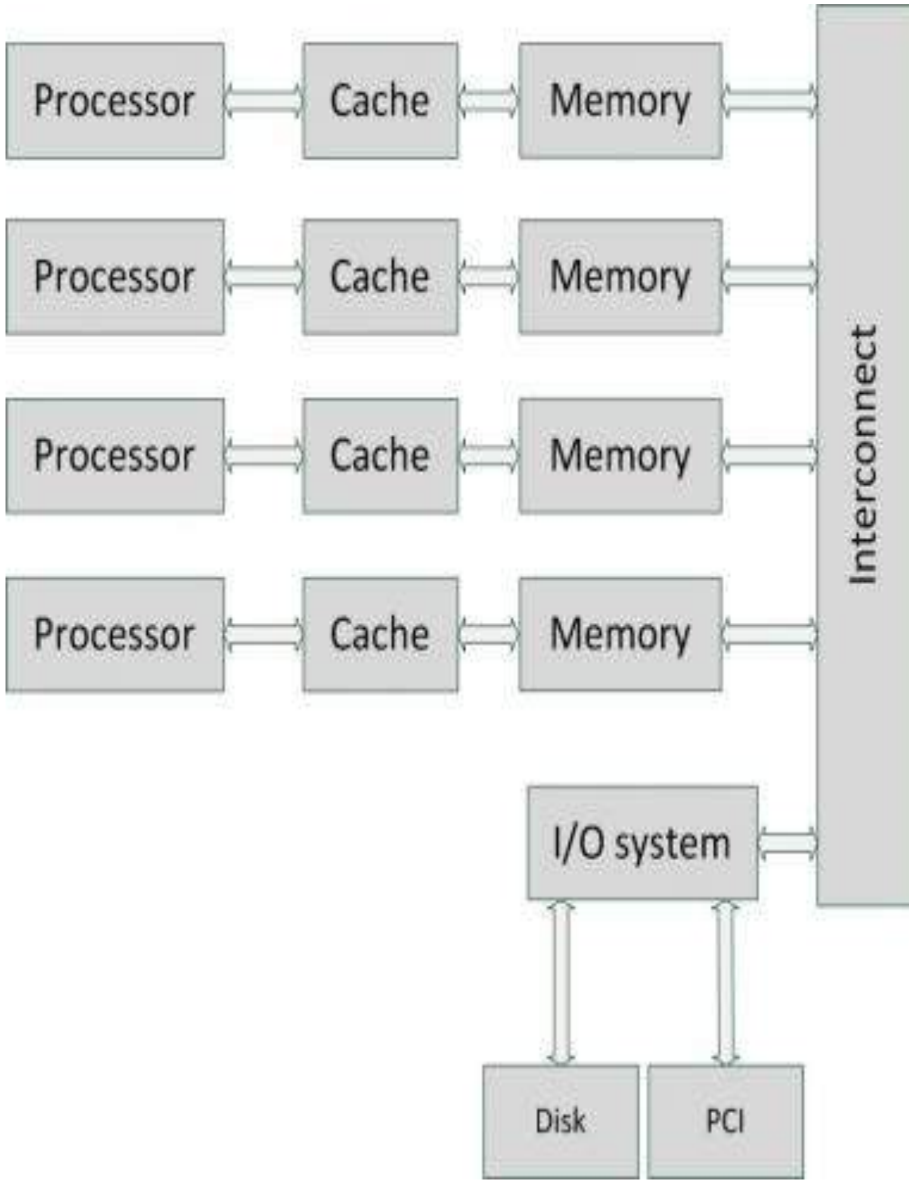
UMA Multiprocessor

- Each processor has the same access time to every memory module whatever the distance to memory module → **local memory access time = shared memory access time**
- Each processor could have a local memory and a cache to reduce traffic to shared memory
- There are three types of UMA architectures:
  - UMA using bus-based SMP: simplest and limited by the bandwidth of bus
  - UMA using crossbar switches
  - UMA using multistage interconnection networks



NUMA Multiprocessor

- NUMA provide a single address space across all the processors but the memory access time depends on the memory location relative to the processor → access processors own local memory is faster than non-local memory
- The first commercial implementation of a NUMA-based Unix system was the Symmetrical Multi Processing XPS-100 family of servers
- NUMA could be viewed as a tightly coupled form of cluster
- NUMA could be split into 2 sub models:
  - NC-NUMA: no cache exists
  - CC-NUMA: there are coherent caches



CC-NUMA Multiprocessor

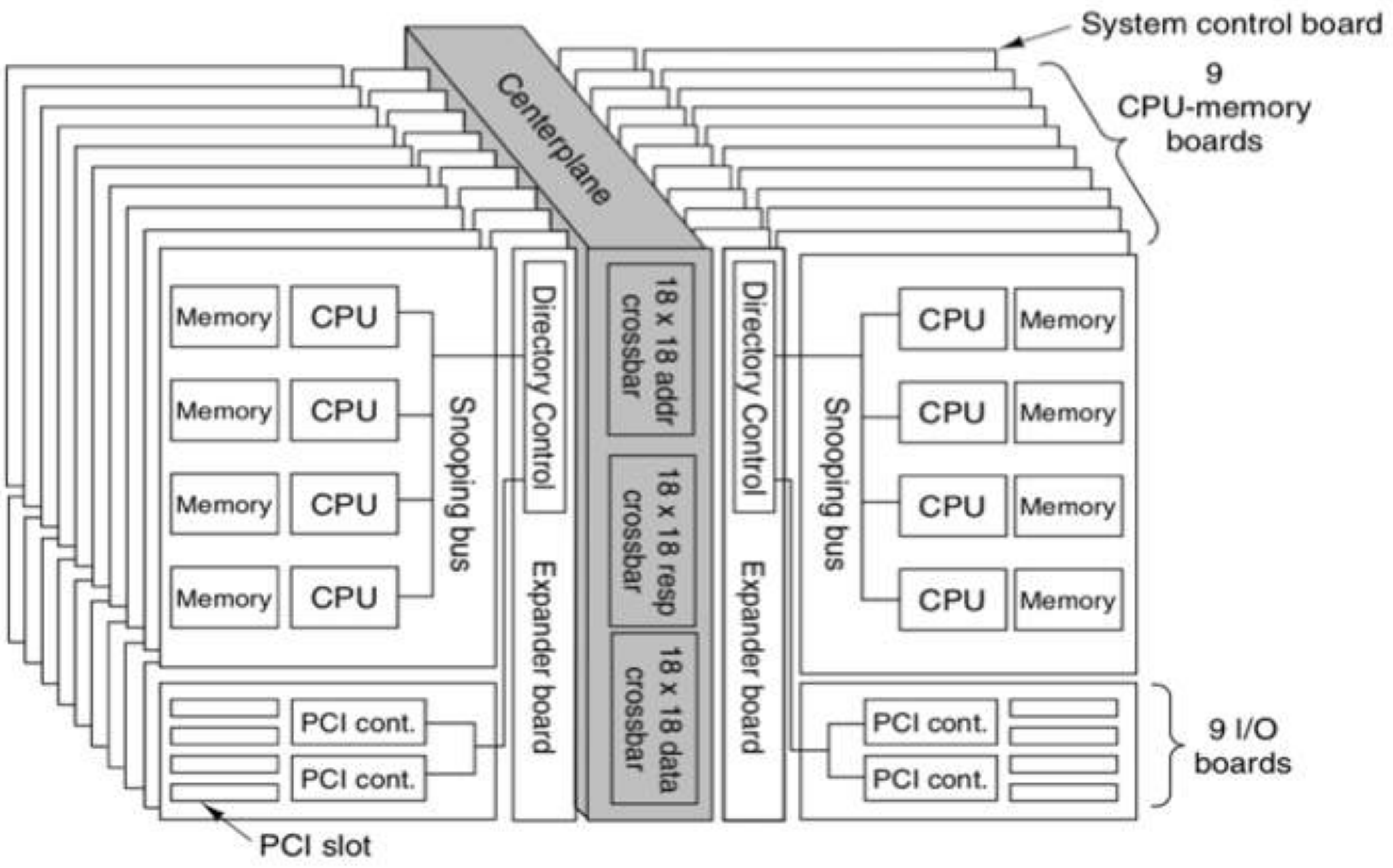
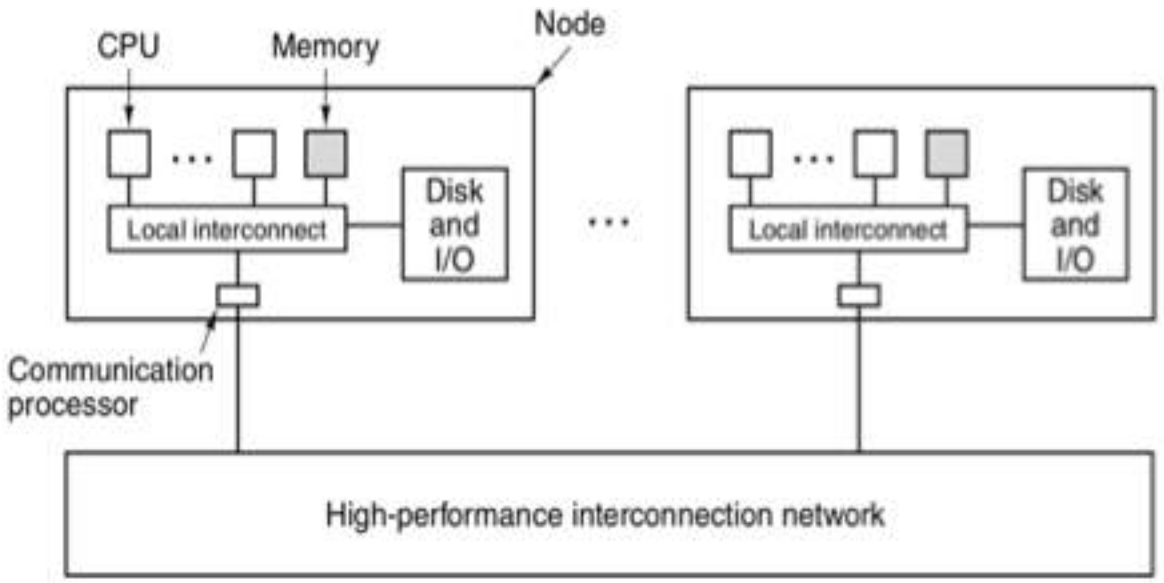
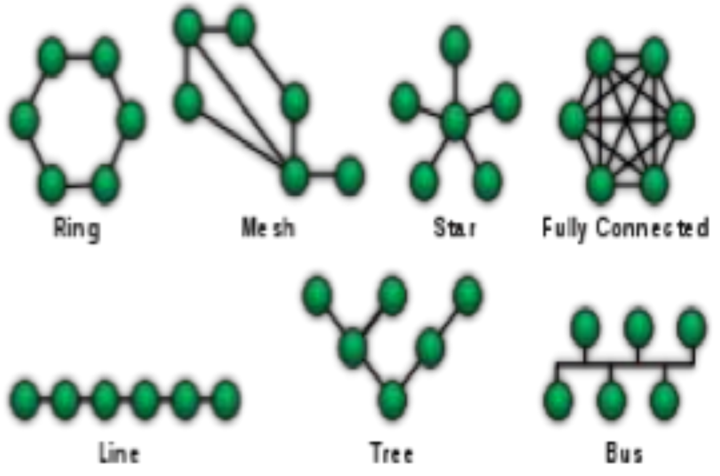


Figure 8-34. The Sun Microsystems E25K multiprocessor.

Cluster: Computer Networking

- **Tightly Multiprocessor limitations**: could not scale to large sizes and memory contention could severely affect the performance
- There is a great deal of interest in building and using parallel computers in which each CPU has its own private memory, not directly accessible to any other CPU → **Multicomputer or Cluster of Multiprocessor**
- In Cluster architecture:
  - Each node in a multicomputer consists of one or a few CPUs, some RAM, a disk and/or other I/O devices, and a communication processor.
  - The communication processors are connected by a high speed interconnection network.
  - Many different topologies, switching schemes, and routing algorithms are used.





Question?