Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Unit 5: Advanced Architecture

Dr. D. C. Kiran

Department of Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Syllabus

~~Unit 1: Basic Processor Architecture and Design~~

~~Unit 2: Pipelined Processor and Design~~

~~Unit 3: Memory~~

~~Unit 4: Input/Output Device Design~~

Unit 5: Advanced Architecture

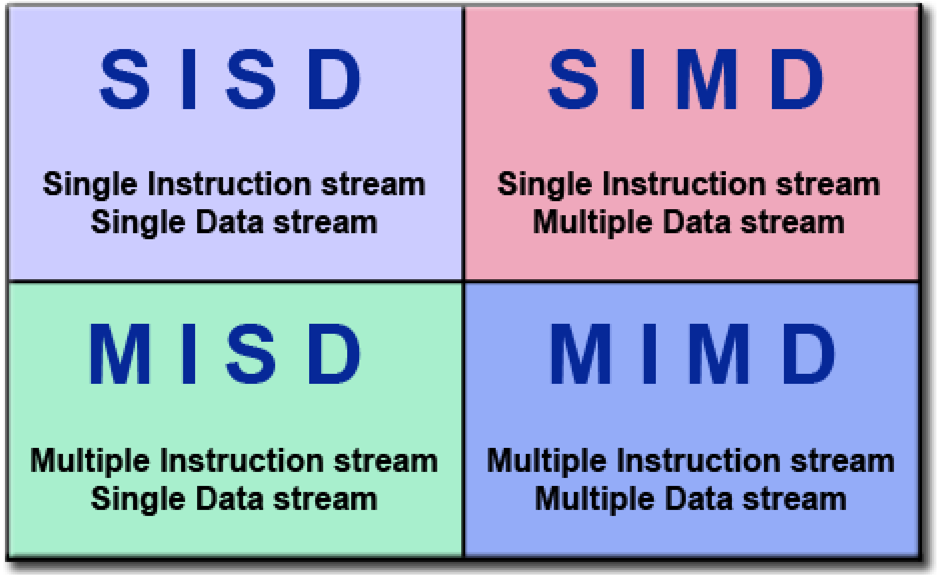
- ~~Need for High Performance Computing~~
- Classification of Parallel Architectures

Flynn's Taxonomy of Computer Architecture

One of the more widely used classifications, in use since 1966

View Computer Architecture in two independent dimensions

- ***Instruction***
- ***Data***



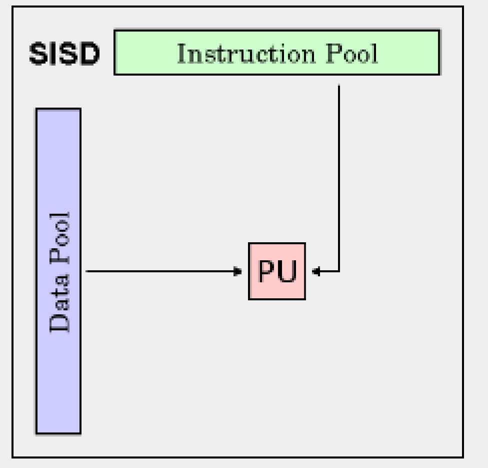
SISD Single Instruction stream Single Data stream	**SIMD** Single Instruction stream Multiple Data stream
MISD Multiple Instruction stream Single Data stream	**MIMD** Multiple Instruction stream Multiple Data stream

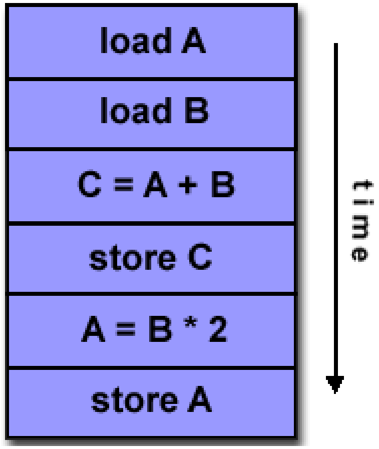
Analogy of Flynn’s Classifications

- An analogy of Flynn’s classification is the check-in desk at an airport
  - SISD: A single desk.
  - SIMD: Many desks and a supervisor with a megaphone giving instructions that every desk obeys.
  - MIMD: Many desks working at their own pace, synchronized through a central database.

SISD: Single Instruction Single Data

- Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle.
- Single data: only one data stream is being used as input during any one clock cycle.
- Deterministic Execution .
- Very few are found in recent days.
- **Few representatives are Intel Atom Family** (Silverthorne, Lincroft, Diamondville, Pineview)




load A

load B

C = A + B

store C

A = B * 2

store A

SIMD: Single Instruction Multiple Data

A type of Parallel computer

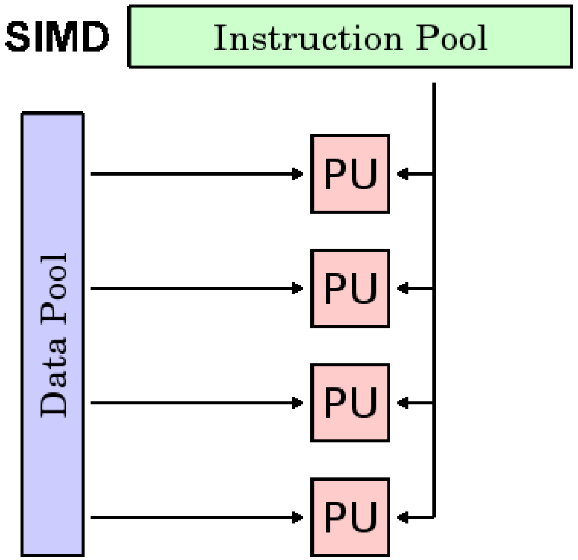
Best suited for specialized problems characterized by a high degree of regularity, such as image processing.

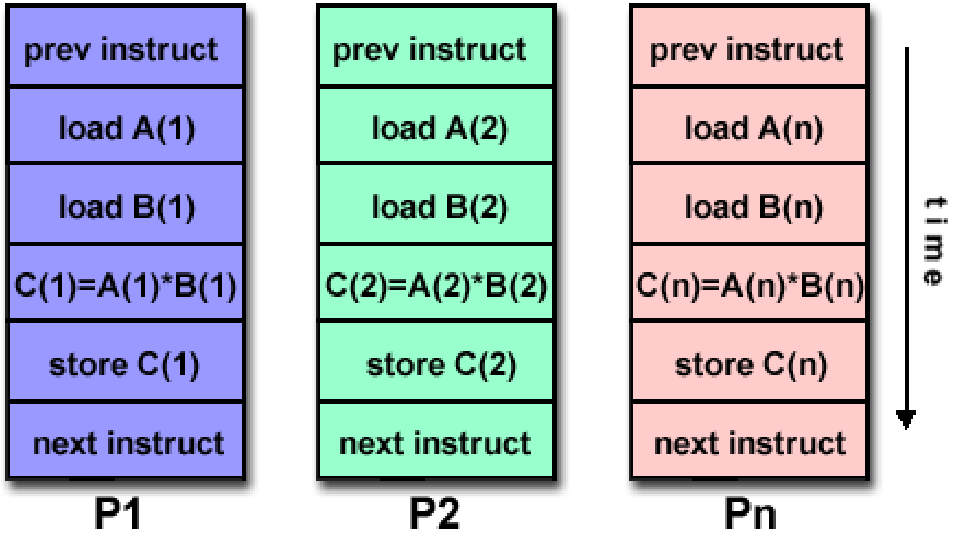
Two varieties: Processor Arrays and Vector Pipelines

Representatives:

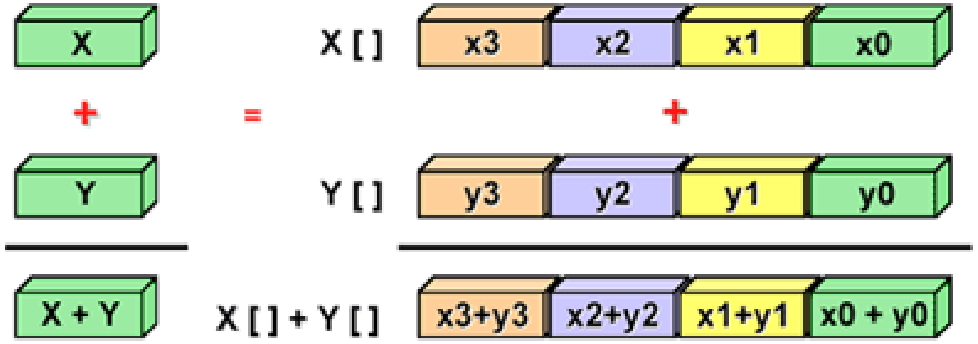
Processor Arrays: Connection Machine CM-2, Maspar MP-1, MP-2

Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820

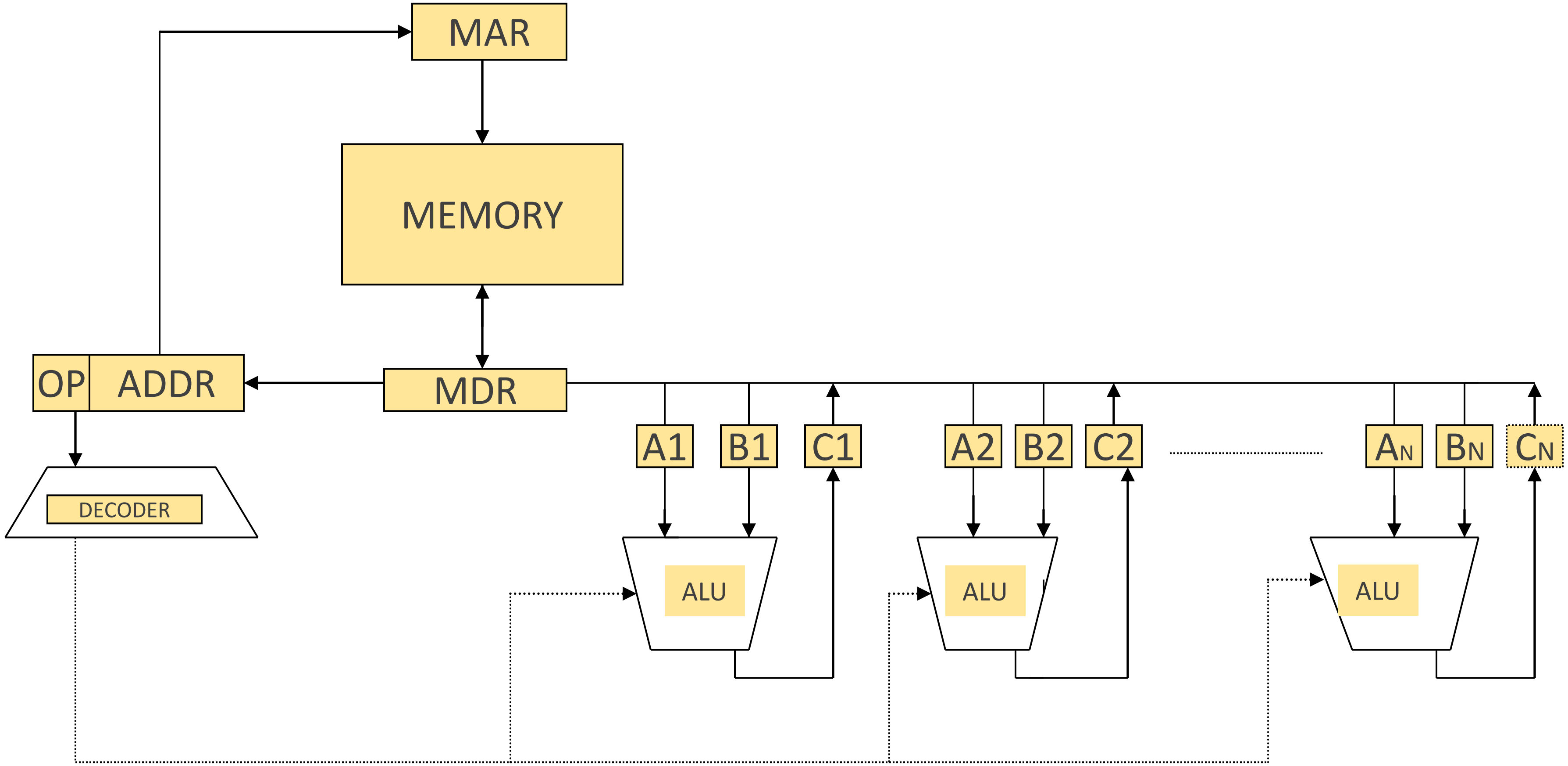




P1	P2	Pn
prev instruct	prev instruct	prev instruct
load A(1)	load A(2)	load A(n)
load B(1)	load B(2)	load B(n)
C(1)=A(1)*B(1)	C(2)=A(2)*B(2)	C(n)=A(n)*B(n)
store C(1)	store C(2)	store C(n)
next instruct	next instruct	next instruct



Array processor (SIMD)

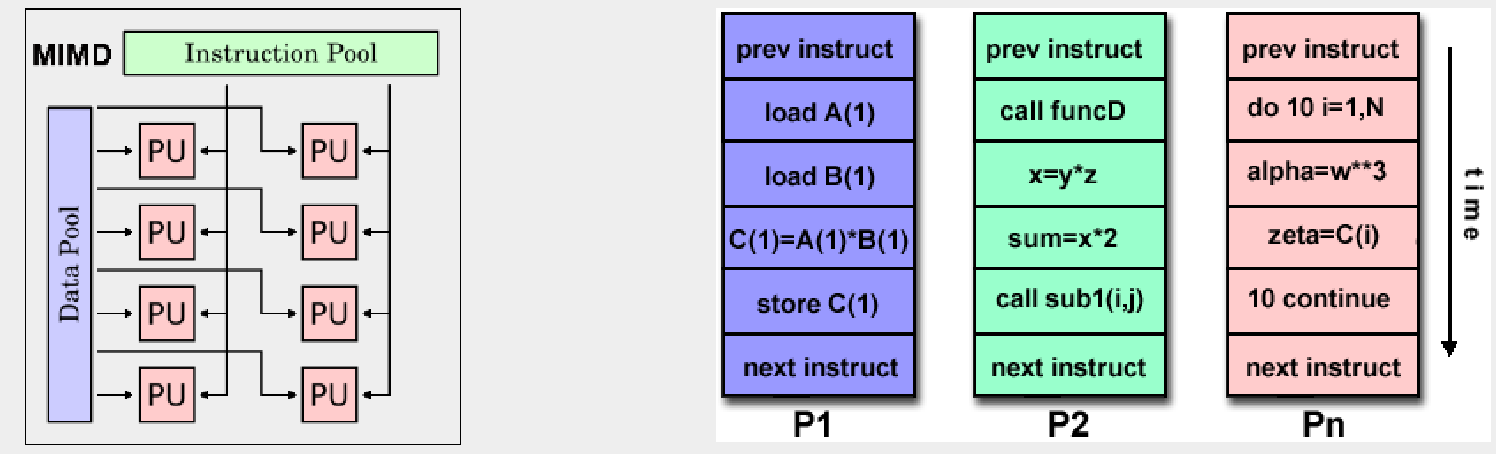


MIMD: Multiple Instruction Multiple Data

- Currently, the most common type of parallel computer. Most modern computers fall into this category.

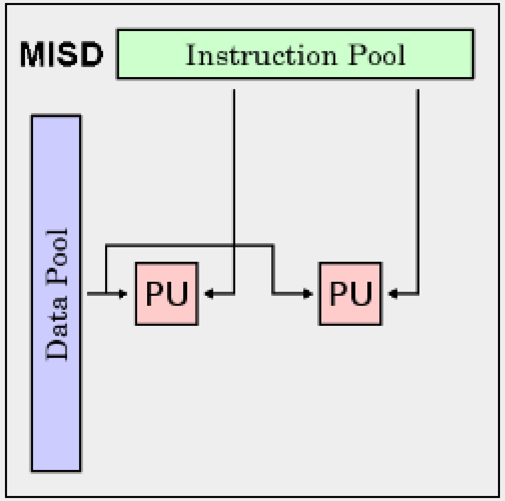
- Execution can be synchronous or asynchronous, deterministic or non-deterministic

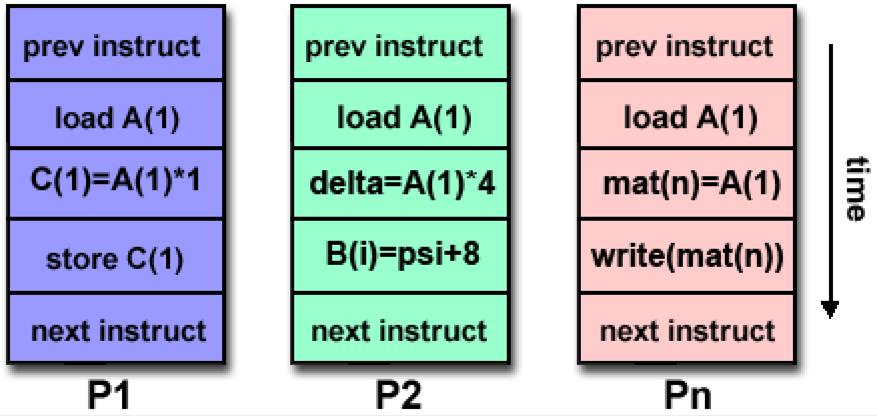
- Representatives: Most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.



MISD: Multiple Instruction Single Data (Does Not exist)

- Few actual examples of this class of parallel computer have ever existed. One is the experimental Carnegie-Mellon University.
- A single data stream is fed into multiple processing units.
- Representatives: Systolic Arrays

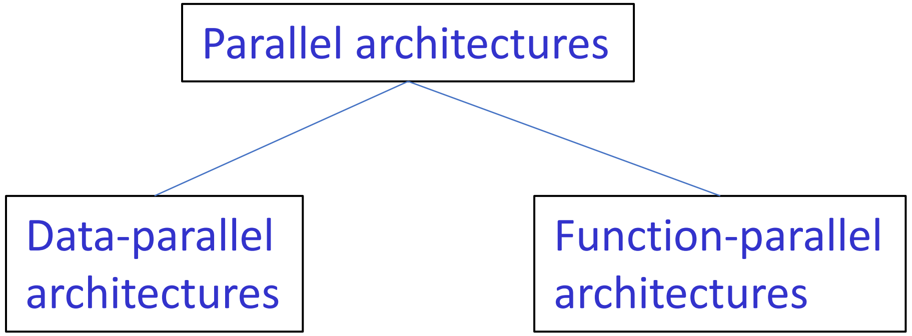




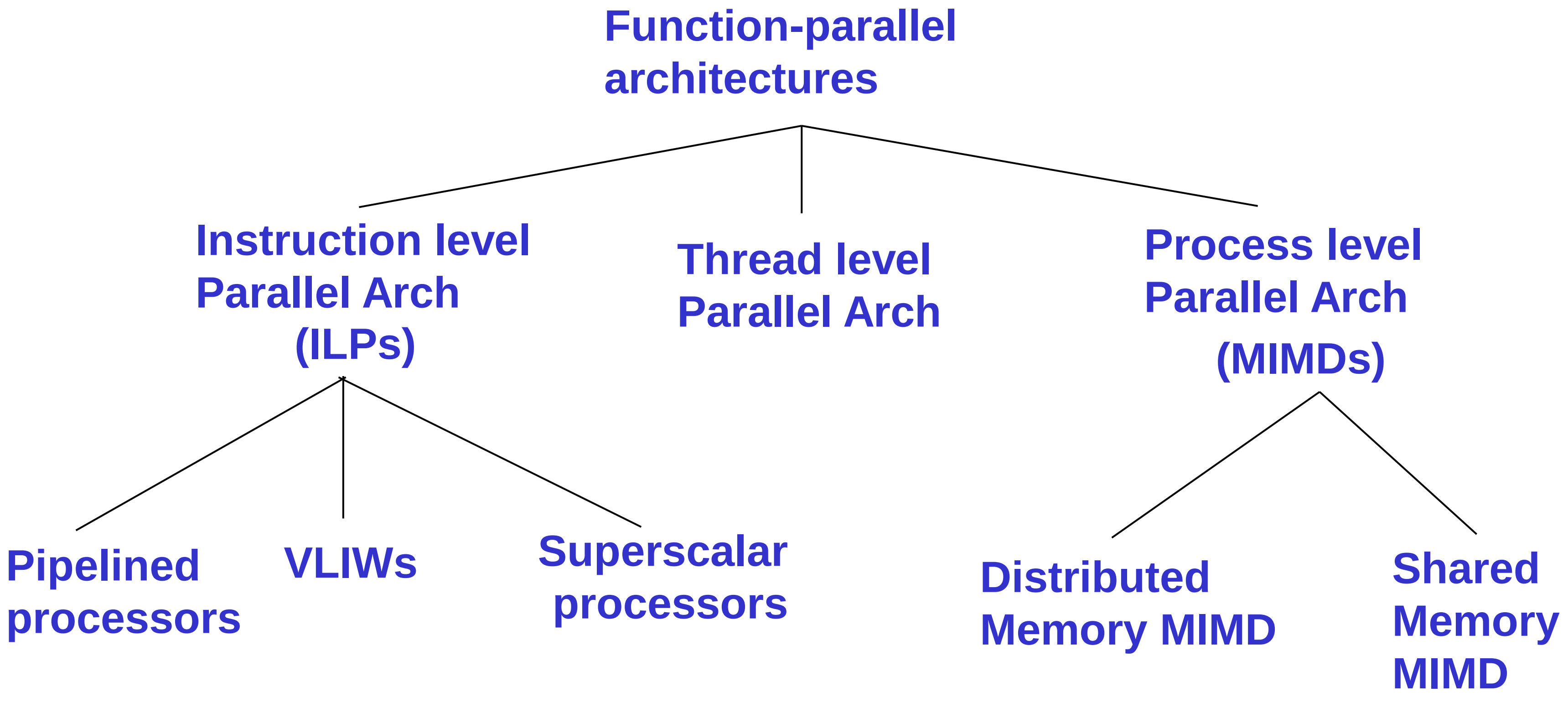
P1	P2	Pn
prev instruct	prev instruct	prev instruct
load A(1)	load A(1)	load A(1)
C(1)=A(1)*1	delta=A(1)*4	mat(n)=A(1)
store C(1)	B(i)=psi+8	write(mat(n))
next instruct	next instruct	next instruct

Modern classification: (Sima, Fountain, Kacsuk)

- Classify based on how parallelism is achieved
  - by operating on multiple data: data parallelism
  - by performing many functions in parallel: function parallelism
    - Control parallelism, task parallelism depending on the level of the functional parallelism.



Functional Parallel Architectures



Function-parallel architectures

Instruction level Parallel Arch (ILPs)

Thread level Parallel Arch

Process level Parallel Arch (MIMDs)

Pipelined processors

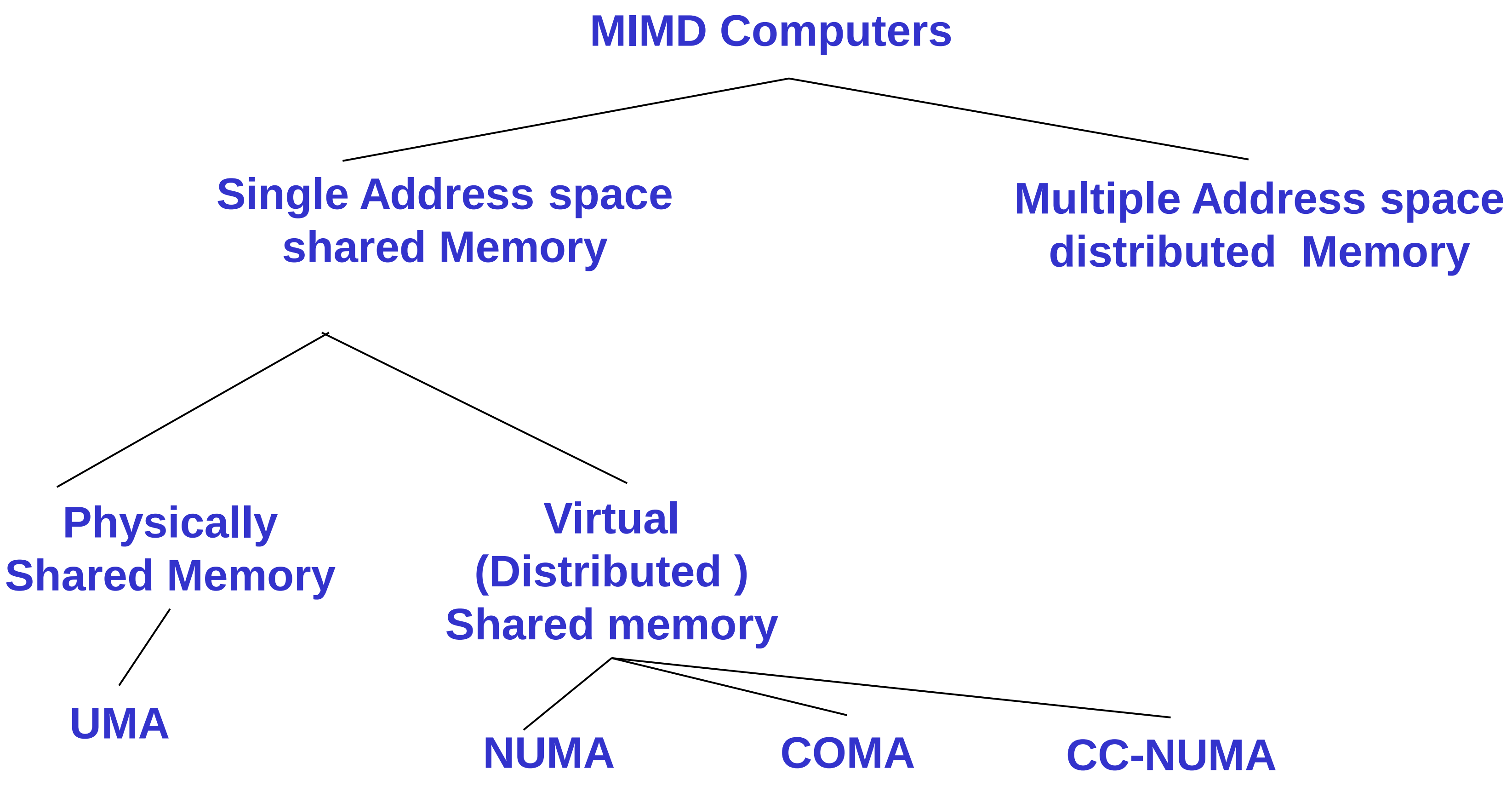
VLIWs

Superscalar processors

Distributed Memory MIMD

Shared Memory MIMD

Classification of MIMD computers


MIMD Computers

Single Address space shared Memory

Multiple Address space distributed Memory

Physically Shared Memory

Virtual (Distributed) Shared memory

UMA

NUMA

COMA

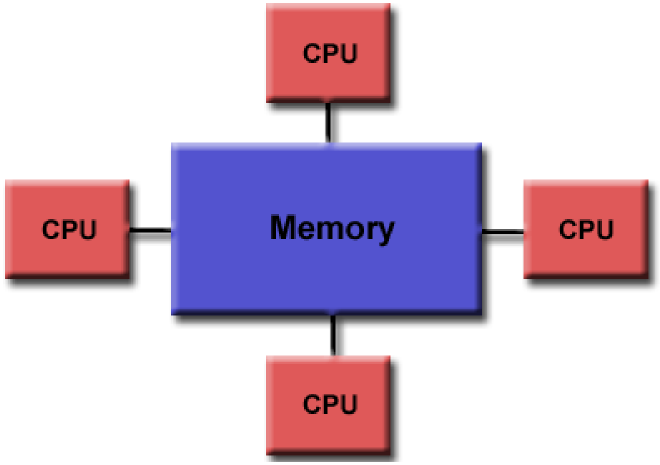
CC-NUMA

Parallel Computer Memory Architectures

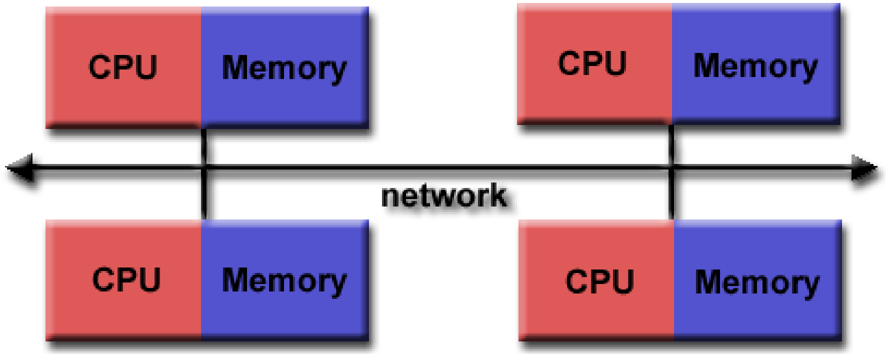
- Shared Memory
- Distributed Memory
- Hybrid Distributed-Shared Memory

Microprocessor & Computer Architecture (μpCA)

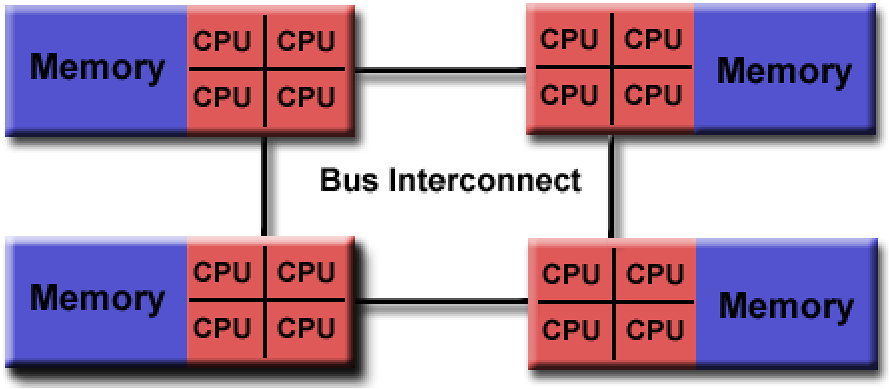
Parallel Computer Memory Architectures



Shared Memory Architecture

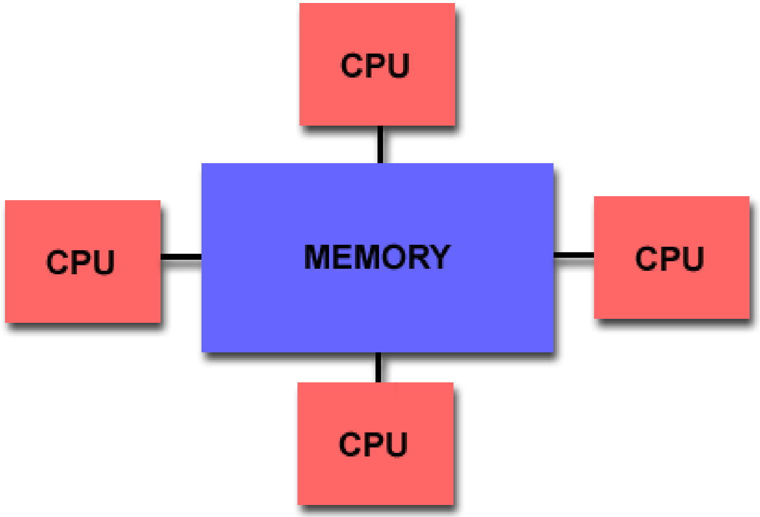


Distributed Memory Architecture



Hybrid Architecture

Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.



Multiple processors can operate independently but share the same memory resources.

Changes in a memory location effected by one processor are visible to all other processors.

Shared memory machines can be divided into two main classes based upon memory access times: ***UMA*** and ***NUMA***.

Shared Memory : UMA vs. NUMA

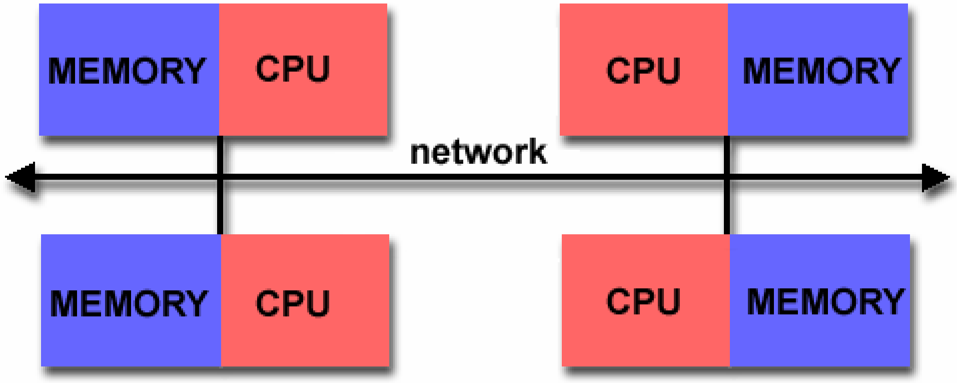
- Uniform Memory Access (UMA):
  - Most commonly represented today by Symmetric Multiprocessor (SMP) machines
  - Identical processors
  - Equal access and access times to memory
  - Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.
- Non-Uniform Memory Access (NUMA):
  - Often made by physically linking two or more SMPs
  - One SMP can directly access memory of another SMP
  - Not all processors have equal access time to all memories
  - Memory access across link is slower
  - If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

Shared Memory: Pro and Con

- **Advantages**
- Global address space provides a user-friendly programming perspective to memory
- Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs
- **Disadvantages:**
- Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increases traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.
- Programmer responsibility for synchronization constructs that insure "correct" access of global memory.
- Expense: it becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing numbers of processors.

Distributed Memory

- Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory.
- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.
- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.
- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.
- The network "fabric" used for data transfer varies widely, though it can can be as simple as Ethernet.

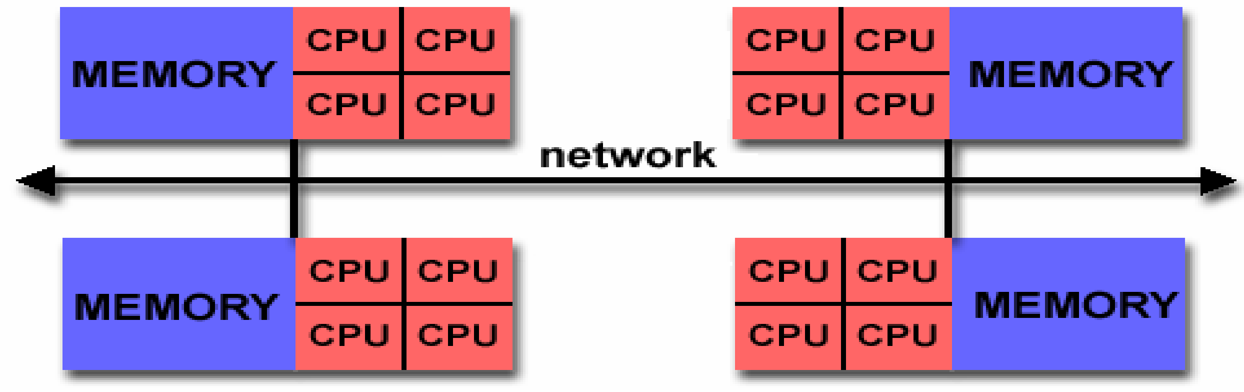


Distributed Memory: Pro and Con

- Advantages
  - Memory is scalable with number of processors. Increase the number of processors and the size of memory increases proportionately.
  - Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency.
  - Cost effectiveness: can use commodity, off-the-shelf processors and networking.
- Disadvantages
  - The programmer is responsible for many of the details associated with data communication between processors.
  - It may be difficult to map existing data structures, based on global memory, to this memory organization.
  - Non-uniform memory access (NUMA) times

Hybrid Distributed-Shared Memory

- The largest and fastest computers in the world today employ both shared and distributed memory architectures.



- The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global.
- The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.
- Current trends seem to indicate that this type of memory architecture will continue to prevail and increase at the high end of computing for the foreseeable future.
- Advantages and Disadvantages: whatever is common to both shared and distributed memory architectures.

Parallel Programming Languages

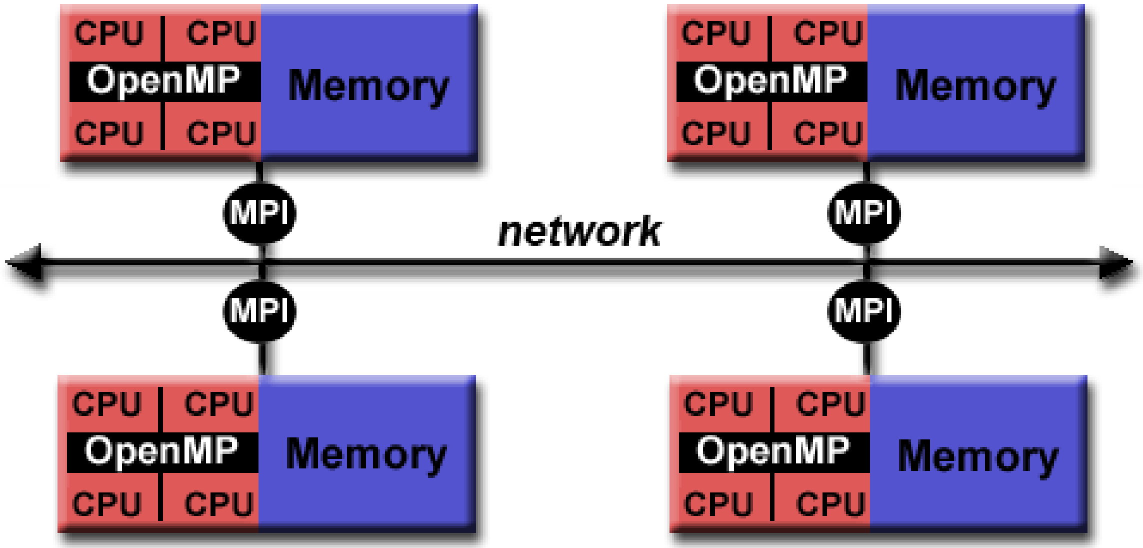
OpenMP: (Open Multi Processing):
API that Support multiprocessing in C, C++, Fortran. Now with Python also.

MPI: (Message Passing Interface):
C, C++, Fortran, Java, Python, Ocaml, R……etc

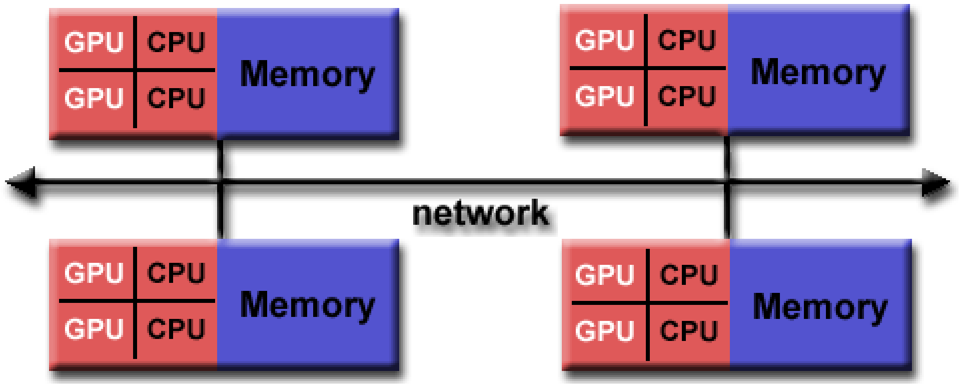
CILK: Customized C Language

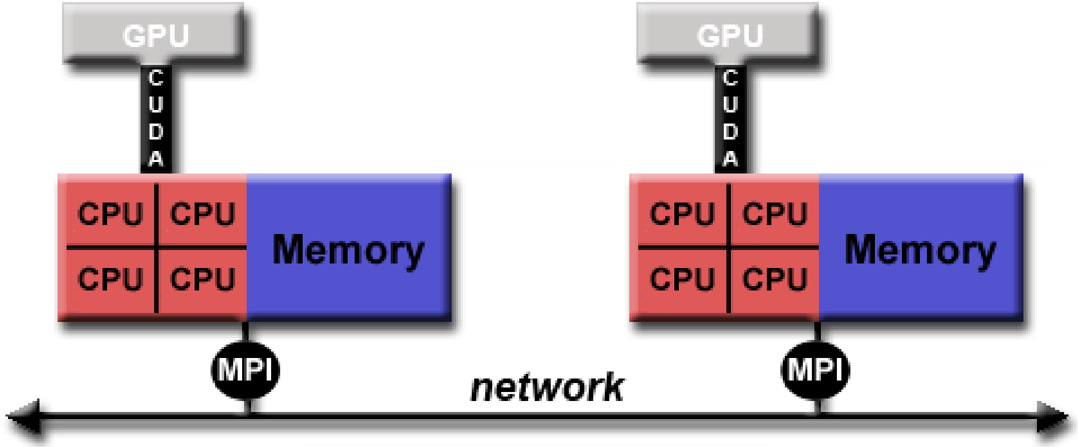
CUDA (Computer Unified Devise Architecture): for Nvidia GPU

Pthreads:



With Graphical Processing Unit





Next Session

Classification of Parallel Computers

THANK YOU

Dr. D. C. Kiran

Department of Computer Science and Engineering

dckiran@pes.edu

9829935135