Overview of Parallel Programming Models

Silvio Stanzani , Raphael Cóbe , Rogério Iope

UNESP - Núcleo de Computação Científica

silvio@ncc.unesp.br , rmcobe@ncc.unesp.br , rogerio@ncc.unesp.br

Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- Hybrid Parallel Architectures
- Memory System
- Vector Processing Units

Agenda

- **Parallelism and Concurrency**
- Parallel Architectures Types
- Hybrid Parallel Architectures
- Memory System
- Vector Processing Units

Parallel Processing

- A parallel computer is a computer system that uses multiple processing elements simultaneously in a cooperative manner to solve a computational problem

- Parallel processing includes techniques and technologies that make it possible to compute in parallel

    - Hardware, networks, operating systems, parallel libraries, languages, compilers, algorithms, tools, …

- Parallel computing is an evolution of serial computing
    - Parallelism is natural
    - Computing problems differ in level / type of parallelism

Concurrency

- Consider multiple tasks to be executed in a computer
- Tasks are concurrent with respect to each if
  - They can execute at the same time (concurrent execution)
  - Implies that there are no dependencies between the tasks
- Dependencies
  - If a task requires results produced by other tasks in order to execute correctly, the task's execution is dependent
  - If two tasks are dependent, they are not concurrent
  - Some form of synchronization must be used to enforce (satisfy) dependencies

Concurrency and Parallelism

- Parallel execution
  - Concurrent tasks actually execute at the same time
  - Multiple (processing) resources have to be available

- Parallelism = concurrency + “parallel” hardware
  - Both are required
  - Find concurrent execution opportunities
  - Develop application to execute in parallel
  - Run application on parallel hardware

Parallelism

- Granularities of parallelism
  - Processes, threads, routines, statements, instructions, …
- These must be supported by hardware resources
  - All aspects of computer architecture offer opportunities for parallel hardware execution (Processors, cores, Memory, DMA, networks);
- Motivations for parallelism
  - Faster time to solution (response time)
  - Solve bigger computing problems (in same time)
  - Effective use of machine resources
  - Cost efficiencies
  - Overcoming memory constraints
- Serial machines have inherent limitations
  - Processor speed, memory bottlenecks, …

How do you get parallelism in the hardware?

- Instruction-Level Parallelism (ILP)
- Data parallelism
  - Increase amount of data to be operated on at same time
- Processor parallelism
  - Increase number of processors
- Memory system parallelism
  - Increase number of memory units
  - Increase bandwidth to memory
- Communication parallelism
  - Increase amount of interconnection between elements
  - Increase communication bandwidth

Agenda

- Parallelism and Concurrency
- **Parallel Architectures Types**
- Hybrid Parallel Architectures
- Memory System
- Vector Processing Units

Classifying Parallel Systems Flynn’s Taxonomy

- Distinguishes multi-processor computer architectures along the two independent dimensions
  - Instruction and Data
  - Each dimension can have one state: Single or Multiple
- SISD: Single Instruction, Single Data
  - Serial (non-parallel) machine
- SIMD: Single Instruction, Multiple Data
  - Processor arrays and vector machines
- MISD: Multiple Instruction, Single Data (unusual)
- MIMD: Multiple Instruction, Multiple Data
  - Most common parallel computer systems

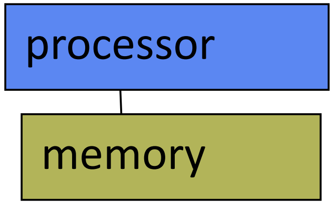
Parallel Architecture Types

- Instruction-Level Parallelism
  - Parallelism captured in instruction processing
- Vector processors
  - Operations on multiple data stored in vector registers
- Shared-memory Multiprocessor (SMP)
  - Multiple processors sharing memory

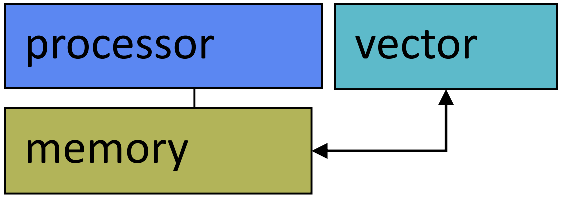
  - Symmetric Multiprocessor (SMP) Multicomputer
  - Multiple computer connect via network
  - Distributed-memory cluster
- Massively Parallel Processor (MPP)

Parallel Architecture Types (2)

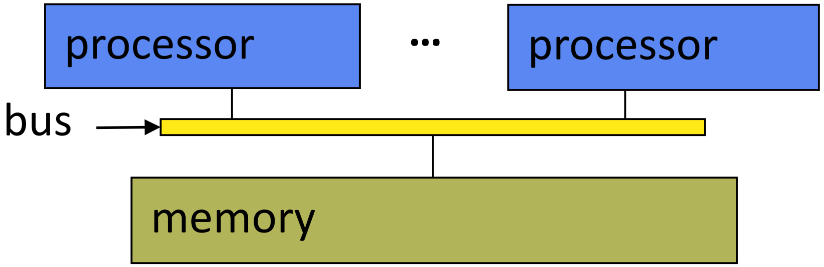
- Uniprocessor
  - Scalar processor



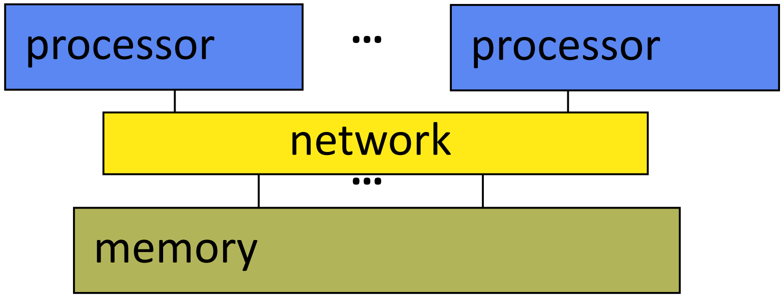
  - Vector processor



- Shared Memory Multiprocessor (SMP)
  - Shared memory address space
  - Bus-based memory system

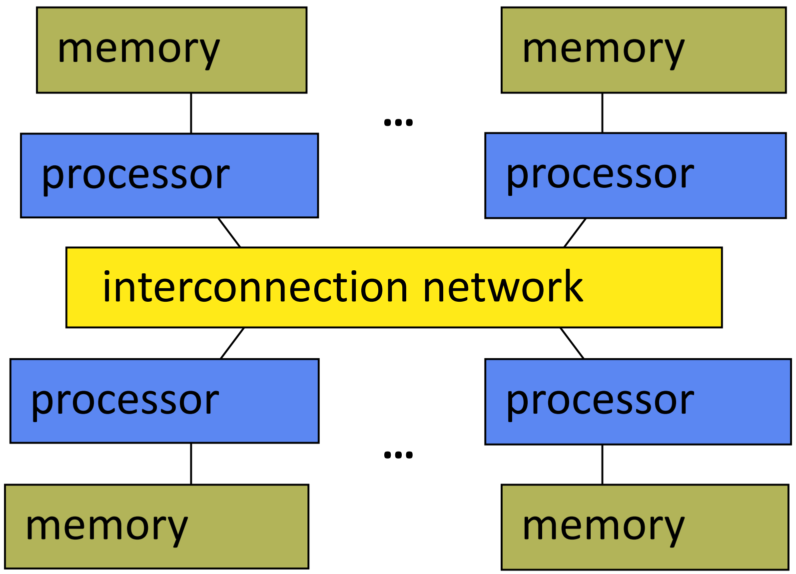


  - Interconnection network



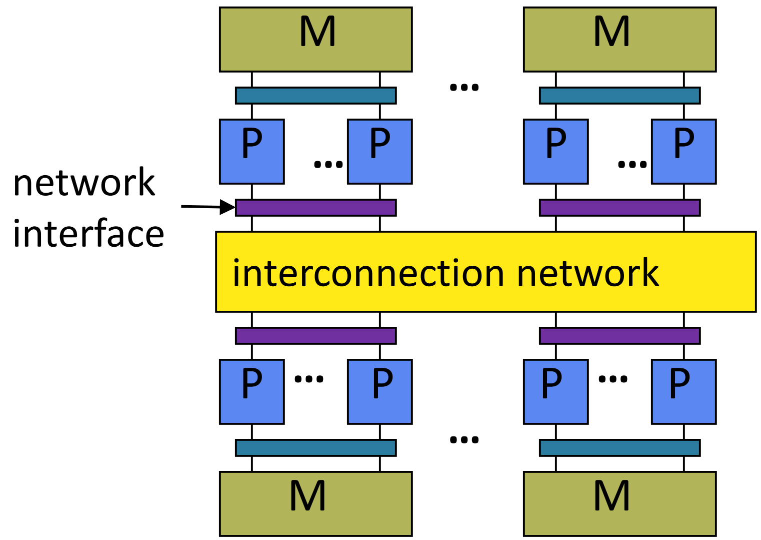
Parallel Architecture Types (3)

- Distributed Memory Multiprocessor
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - ❑ Many, many processors

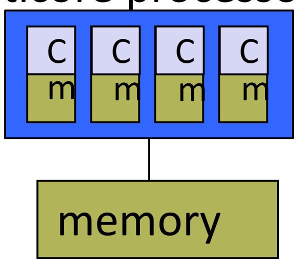
- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



  - Can also be regarded as MPP if processor number is large

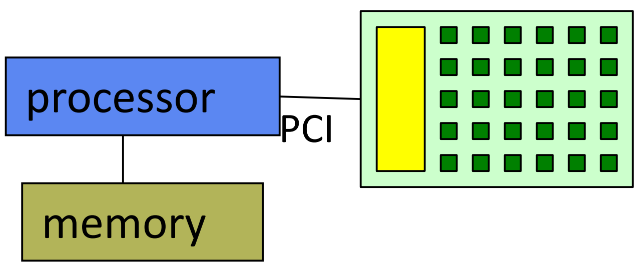
Parallel Architecture Types (4)

- Multicore
  - Multicore processor

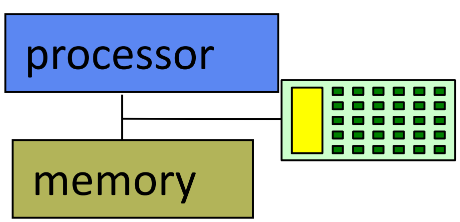


cores can be hardware multithreaded (hyperthread)

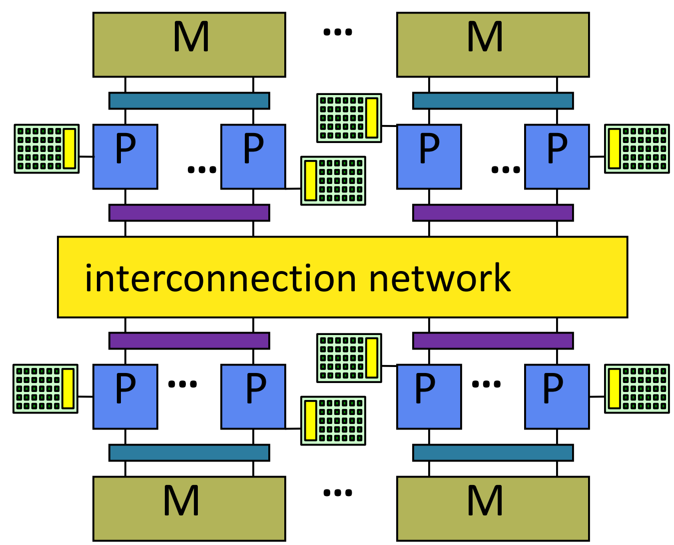
  - Coprocessor



  - "Fused" processor accelerator



- Multicore SMP+coprocessor Cluster
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes
  - Coprocessor attached



Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- **Hybrid Parallel Architectures**
- Memory System
- Vector Processing Units

Hybrid Parallel Architectures

- Heterogeneous computational systems:
  - Multicore processors;
  - Multi-level memory sub-system;
  - Input and Output sub-system;
- Multi-level parallelism:
  - Processing core;
  - Chip multiprocessor;
  - Computing node;
  - Computing cluster;
- Hybrid Parallel architectures
  - Coprocessors and accelerators;

Parallel Architectures

- Heterogeneous computational systems:
  - Multicore processors

Vector Instructions (SIMD)

A7	A6	A5	A4	A3	A2	A1	A0
			+				
B7	B6	B5	B4	B3	B2	B1	B0
			=				
A7+B7	A6+B6	A5+B5	A4+B4	A3+B3	A2+B2	A1+B1	A0+B0

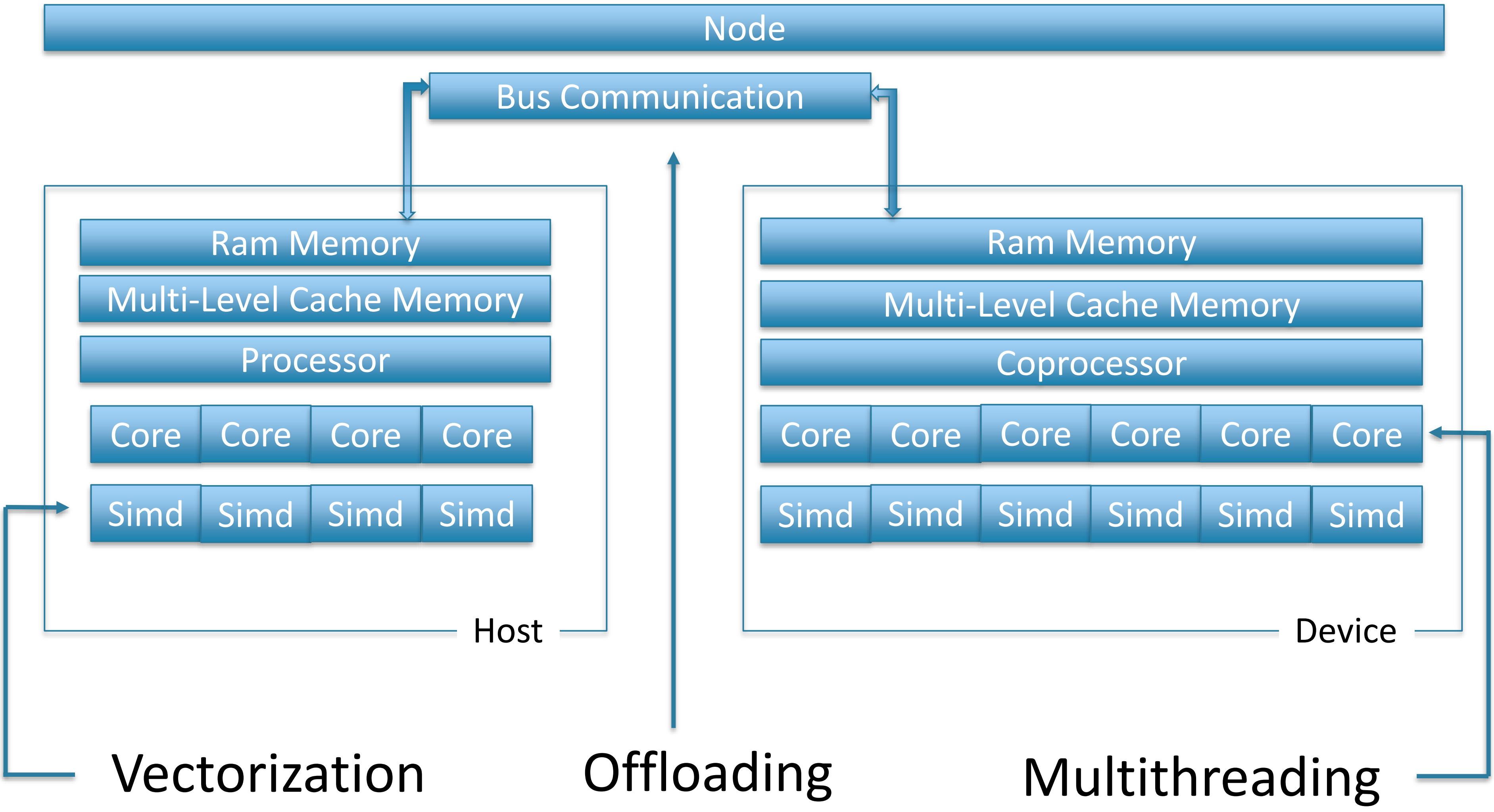
Scalar Instructions

A
+
B
=
A+B

  - Multi-level memory
    - ❑ Ram Memory;
    - ❑ Multi-level Cache.

Processor 1			Processor 2		
Core 1	Core 2	Core N	Core 1	Core 2	Core N
L1	L1	L1	L1	L1	L1
L2	L2	L2	L2	L2	L2
L3			L3		
Ram					

Hybrid Parallel Architectures



Hybrid Parallel Architectures

- Exploring parallelism in hybrid parallel architectures
  - Multithreading
  - Vectorization
    - ❑ Auto vectorization
    - ❑ Semi-auto vectorization
    - ❑ Explicit vectorization
  - Offloading
    - ❑ Offloading code to device

Multi Level Parallelism

Cluster	Group of computers communicating through fast interconnect
Coprocessors/Accelerators	Special compute devices attached to the local node through special interconnect
Node	Group of processors communicating through shared memory
Socket	Group of cores communicating through shared cache
Core	Group of functional units communicating through registers
Hyper-Threads	Group of thread contexts sharing functional units
Superscalar	Group of instructions sharing functional units
Pipeline	Sequence of instructions sharing functional units
Vector	Single instruction using multiple functional units

Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- Hybrid Parallel Architectures
- **Memory System**
- Vector Processing Units

Memory System

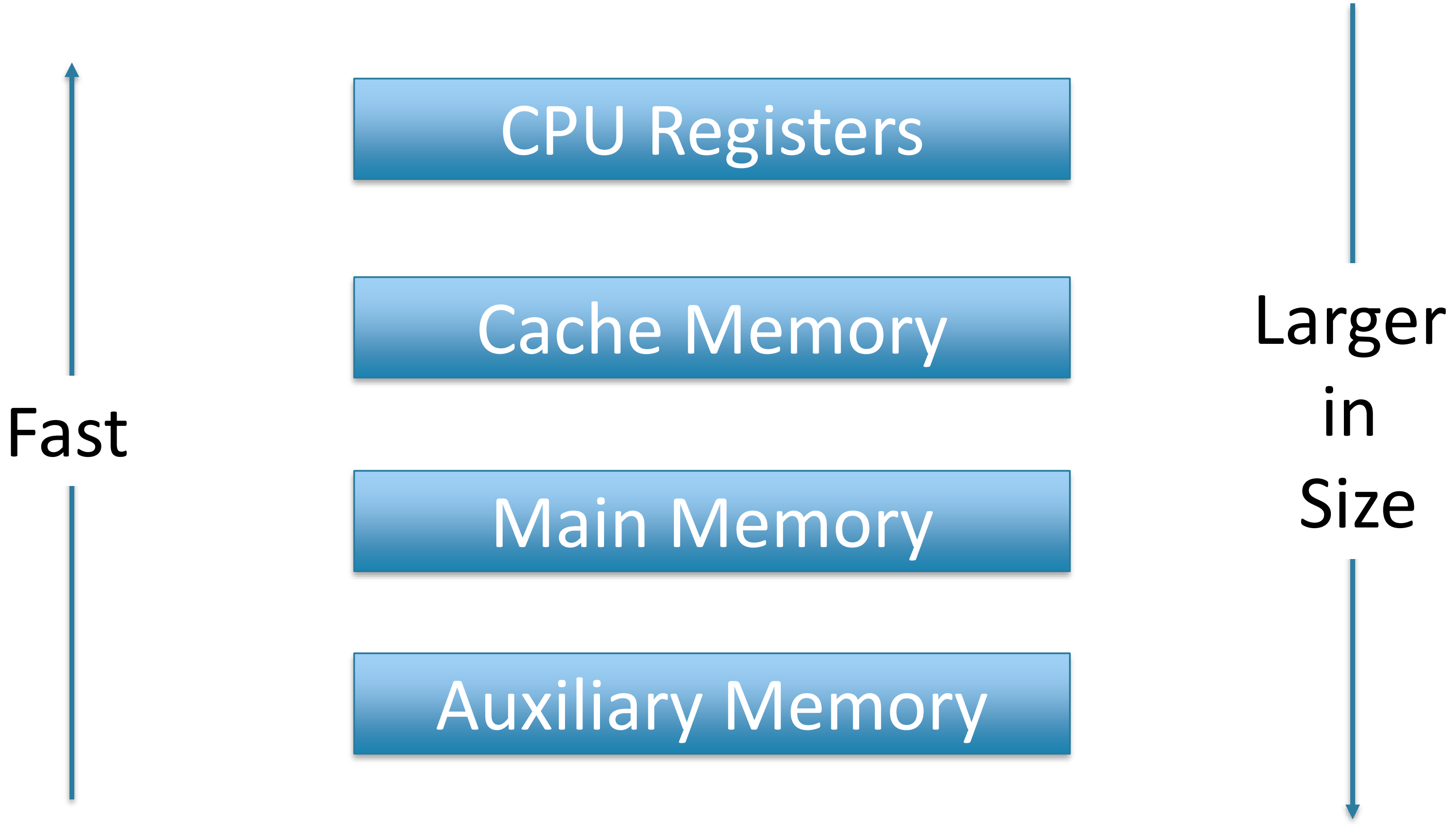
- CPU Register: internal Processor Memory. Stores data or instruction to be executed;

- Cache: stores segments of programs currently being executed in the CPU and temporary data frequently needed in the present calculations;

- Main memory: only program and data currently needed by the processor resides in main memory;

- Auxiliary memory: devices that provides backup storage.

Memory Hierarchy



Cache Memory

- Cache Memory is employed in computer systems to compensate for the difference in speed between main memory access time and processor logic.

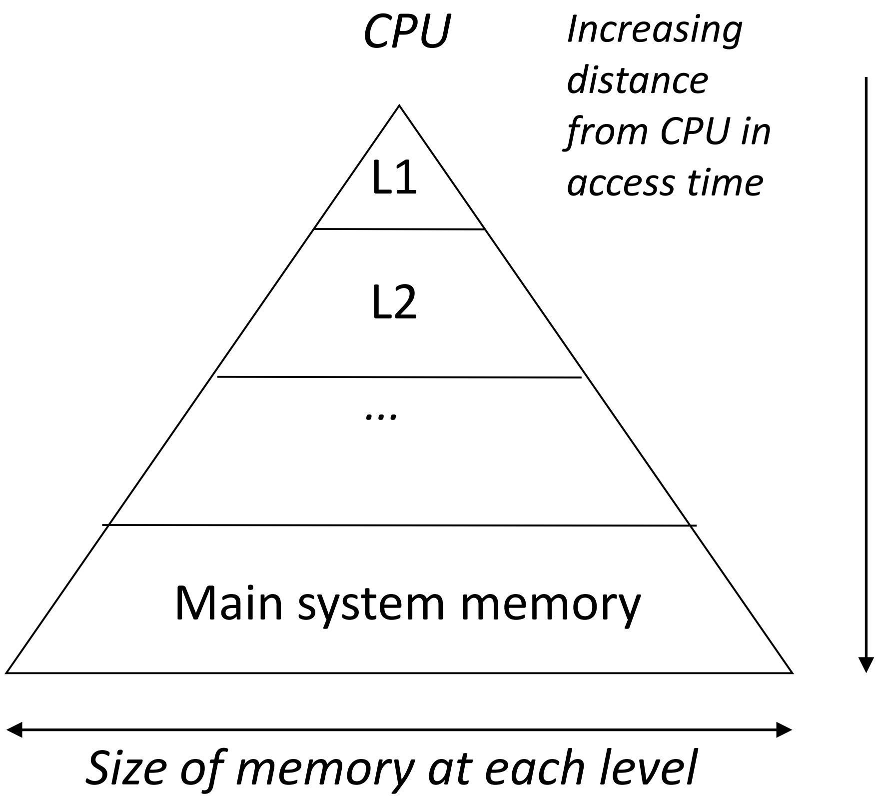
- Operating System controls the load of Data to Cache; such load can be guided by the developer

Cache Memory

- The Performance of cache memory is frequently measured in terms of hit ratio.
  - When the CPU refers to memory and finds the word in cache, it is said to produce a **hit**.
  - If the word is not found in cache, it is in main memory and it counts as a **miss**

Locality

- Temporal locality: if an item was referenced, it will be referenced again soon (e.g. cyclical execution in loops);
- Spatial locality: if an item was referenced, items close to it will be referenced too (the very nature of every program – serial stream of instructions)



Agenda

- Parallelism and Concurrency
- Parallel Architectures Types
- Hybrid Parallel Architectures
- Memory System
- **Vector Processing Units**

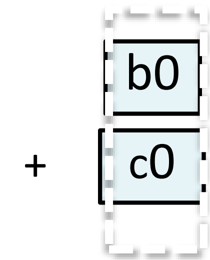
Scalar and Vector Instructions

- Scalar Code computes this one-element at a time.

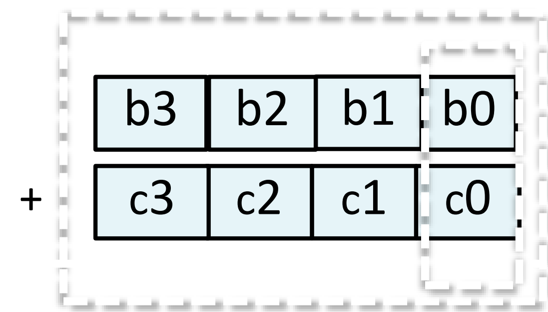
- Vector (or SIMD) Code computes more than one element at a time. SIMD stands for **S**ingle **I**nstruction **M**ultiple **D**ata.

```
float *A, *B, *C;
for(i=0;i<n;i++){
  A[i] = B[i] + C[i];
}
```

- Scalar



- SIMD



Vectorization

- Vectorization
  - Loading data into cache accordingly;
  - Store elements on SIMD registers or vectors;
  - Apply the same operation to a set of Data at the same time;
  - Iterations need to be independent;
  - Usually on inner loops.

Scalar loop

```
for (int i = 0; i < N; i++)
    c[i] = a[i] + b[i];
```

SIMD loop (4 elements)

```
for (int i = 0; i < N; i += 4)
    c[i:4] = a[i:4] + b[i:4];
```