UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES

Compute Models

Heterogeneous Computing

Professor: Dr. Joel Fuentes - jfuentes@ubiobio.cl

Teaching Assistants:

- Daniel López - daniel.lopez1701@alumnos.ubiobio.cl
- Sebastián González - sebastian.gonzalez1801@alumnos.ubiobio.cl

Course website: http://www.face.ubiobio.cl/~jfuentes/classes/ch

Contents

- Flynn's taxonomy
- Types of parallelism
- Programming models

Flynn's taxonomy

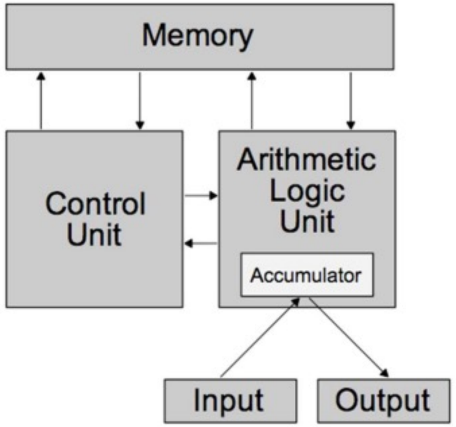
- Taxonomy to classify computer systems by the number of instruction streams and data streams.
- Defined in 1972. Theory still used today.
- It has various restrictions, but at a general level it is useful.

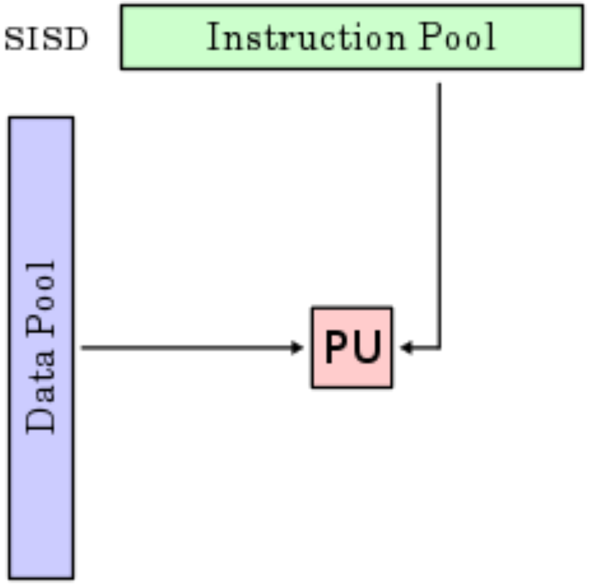
Flynn's taxonomy

- SISD - Single Instruction Single Data
- SIMD - Single Instruction Multiple Data
- MISD - Multiple Instruction Single Data
- MIMD - Multiple Instruction Multiple Data

SISD

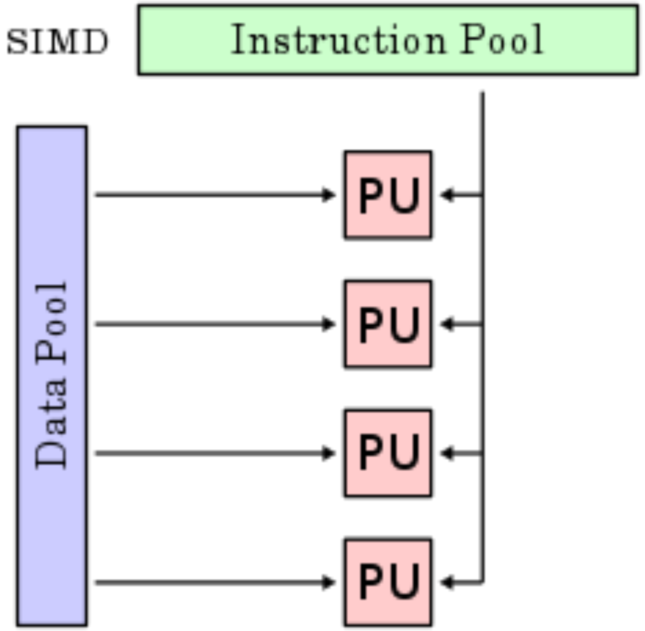
- Single Instruction Single Data
- Corresponds to the architecture of Von Neumann
- Implements the universal Turing machine
- Serial algorithms
- Systems with this computing model execute one instruction at a time on a piece of data.
- Example: Single-core x86 processors





SIMD

- Single Instruction Multiple Data
- Corresponds to parallel computing units that operate on multiple data at once.
- The same instruction is applied on multiple (different) data.
- Vector-based programming.
- Examples: GPUs and AI accelerators.



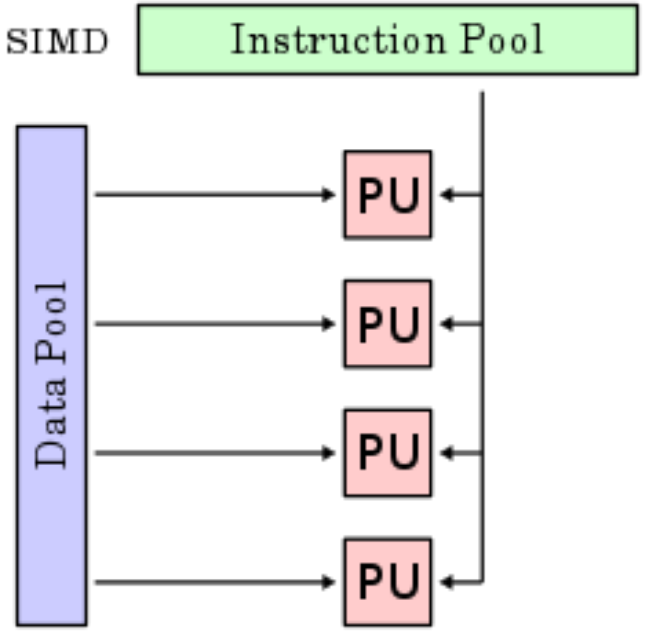
SIMD

- Example of use:

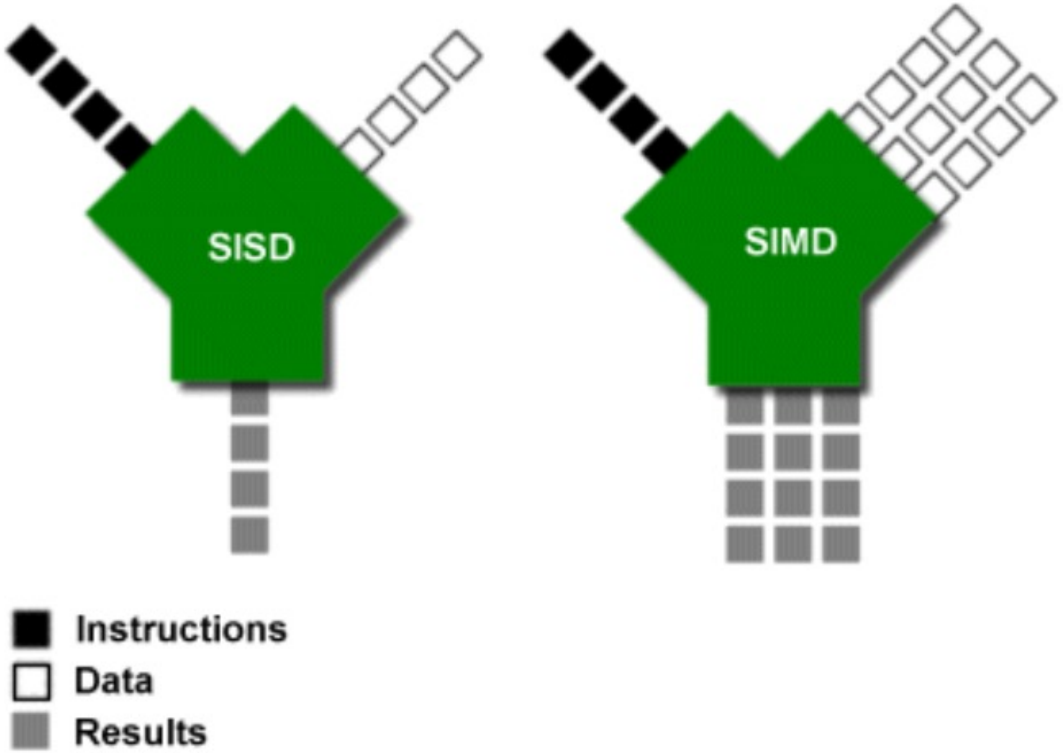
```
for (i = 0; i < n ; i++){
   x[i] += y[i]
}
```

If n processing units (PU) exist and they all execute the same instruction, then the full iteration can be executed by a SIMD instruction.

SIMD is very efficient at solving massive problems in data and vectors/arrays.

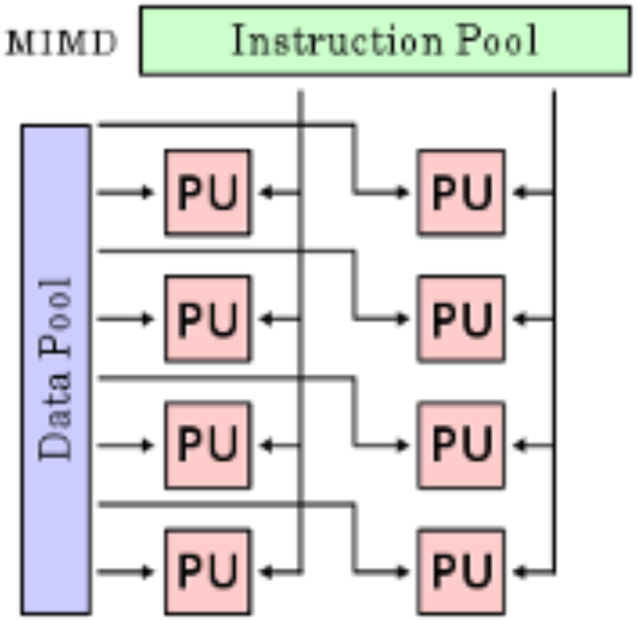


SISD vs SIMD



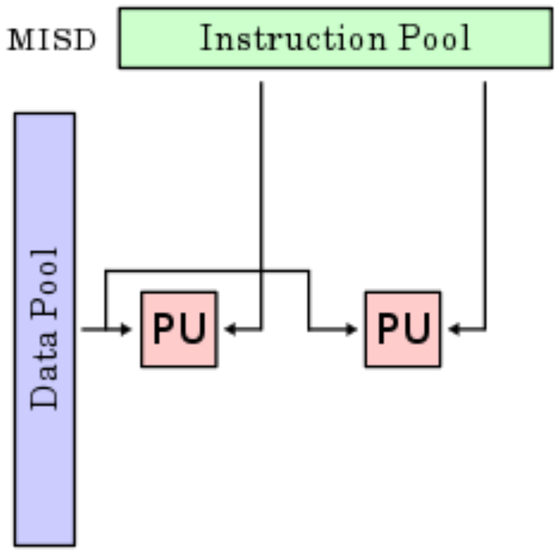
MIMD

- Multiple Instructions Multiple Data
- Supports simultaneous instruction flows operating on multiple data streams.
- Corresponds to multi-core computer systems, clusters, ccNUMA, etc.
- Usually each PU is executed asynchronously. There is no global clock signal.
- Can be implemented in shared memory system or distributed memory.

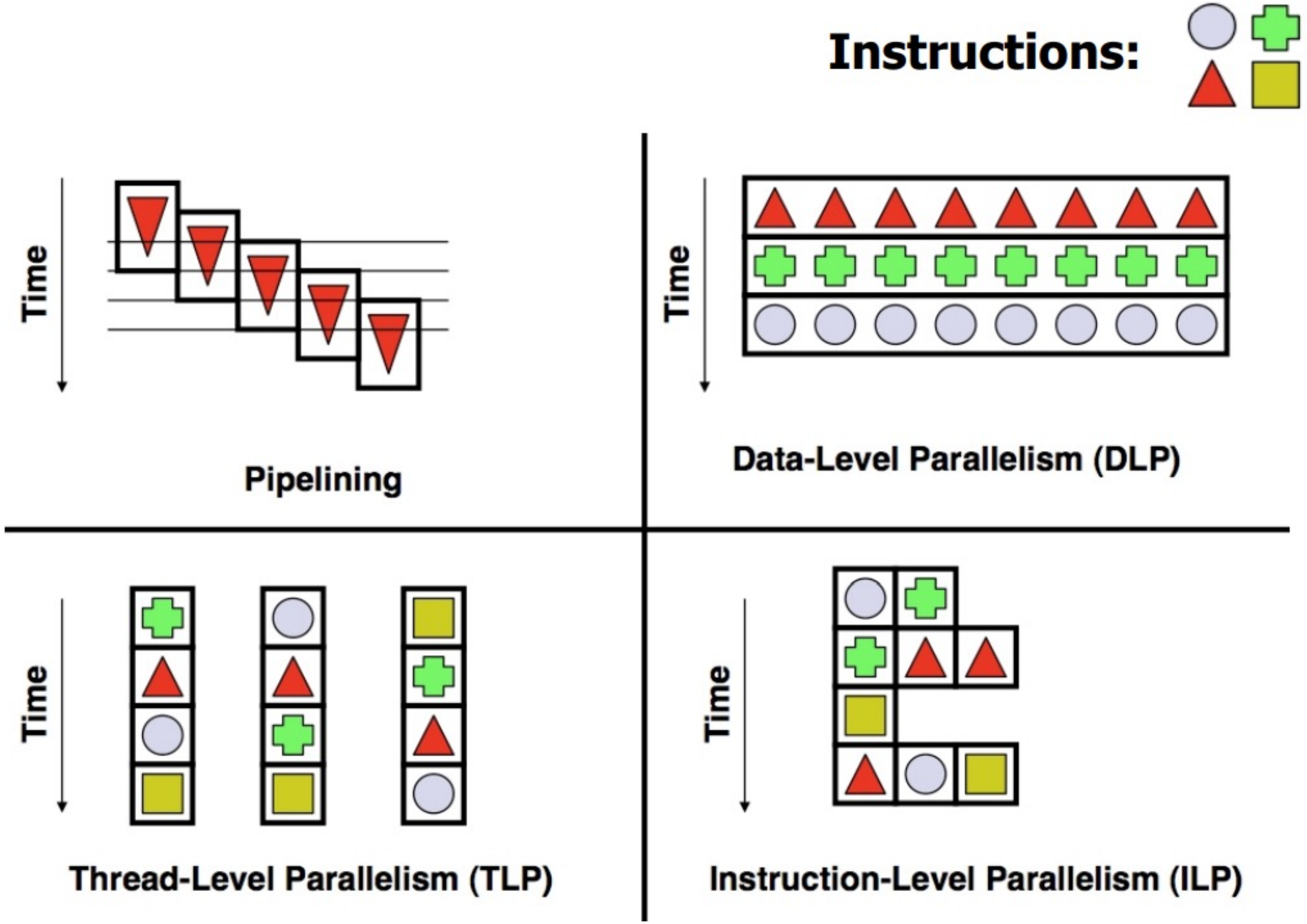


MISD

- Multiple Instructions Single Data
- Supports simultaneous instruction flows operating on a piece of data.
- Multiple PUs operate independently over the same data stream.
- Difficult and unviable implementation.
- Example: Systems that can be used to detect and correct errors.



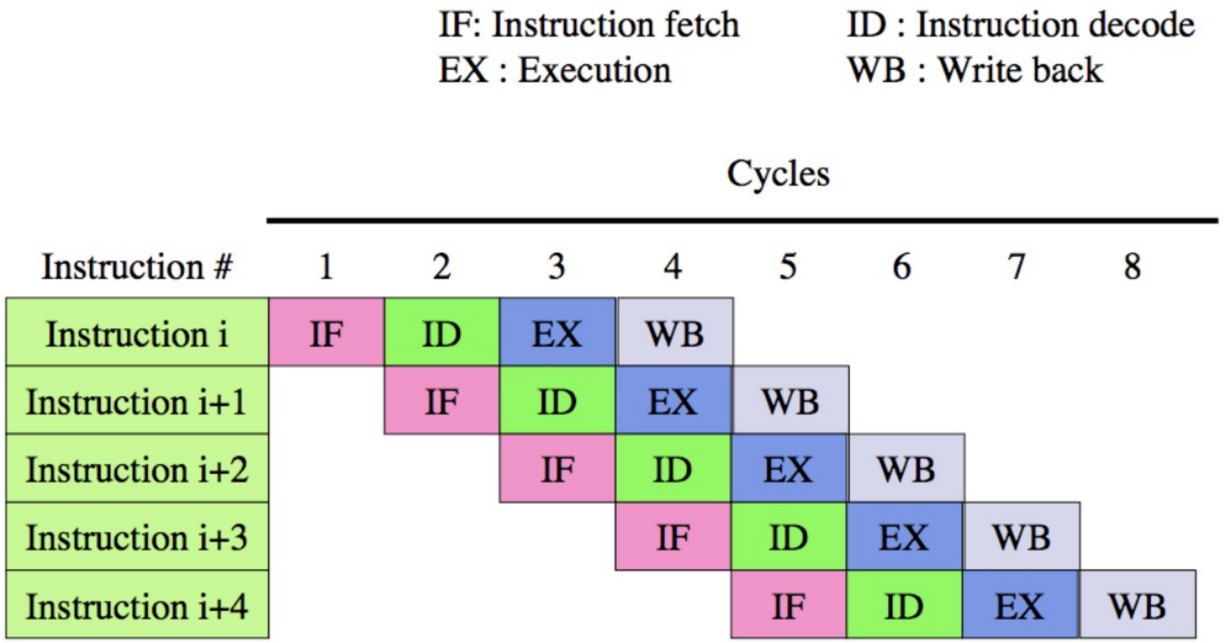
Types of Parallelism


Instructions:
Time
Pipelining
Time
Data-Level Parallelism (DLP)
Time
Thread-Level Parallelism (TLP)
Time
Instruction-Level Parallelism (ILP)

Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

Pipelining

- Corresponds to SISD architecture



IF: Instruction fetch
EX : Execution

ID : Instruction decode
WB : Write back

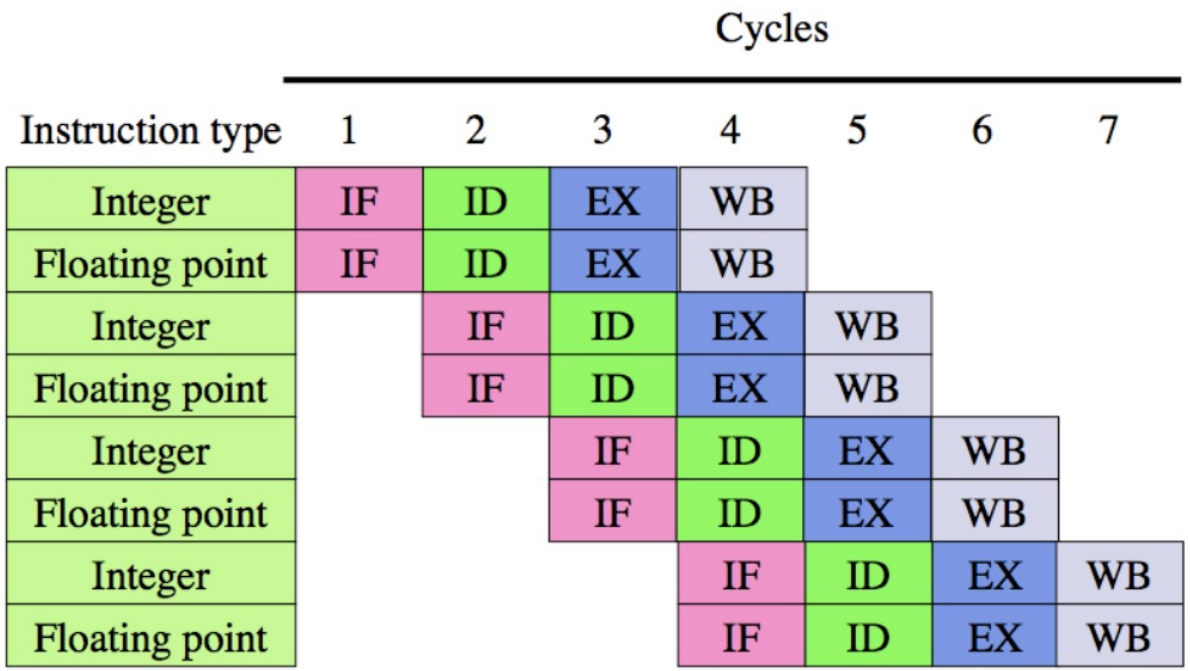
Cycles

Instruction #	1	2	3	4	5	6	7	8
Instruction i	IF	ID	EX	WB				
Instruction i+1		IF	ID	EX	WB			
Instruction i+2			IF	ID	EX	WB		
Instruction i+3				IF	ID	EX	WB	
Instruction i+4					IF	ID	EX	WB

Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

Instruction-level parallelism (ILP)

- Corresponds to SISD architecture

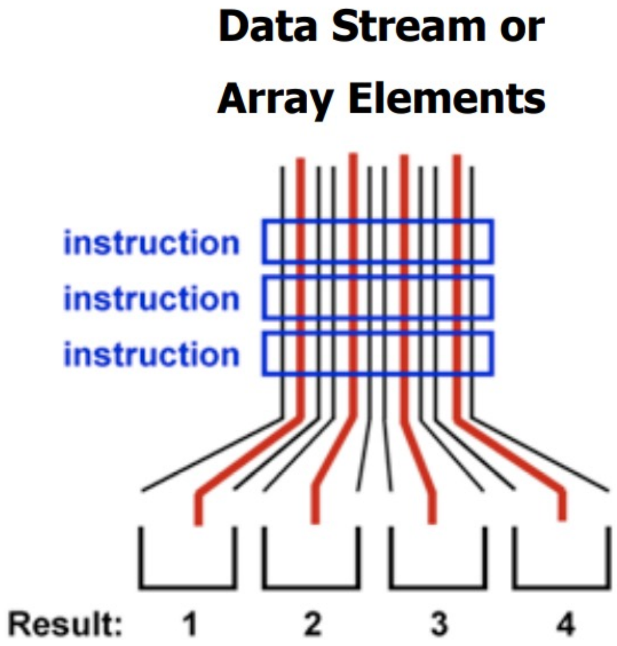


	Cycles						
Instruction type	1	2	3	4	5	6	7
Integer	IF	ID	EX	WB			
Floating point	IF	ID	EX	WB			
Integer		IF	ID	EX	WB		
Floating point		IF	ID	EX	WB		
Integer			IF	ID	EX	WB	
Floating point			IF	ID	EX	WB	
Integer				IF	ID	EX	WB
Floating point				IF	ID	EX	WB

Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

Data-level parallelism

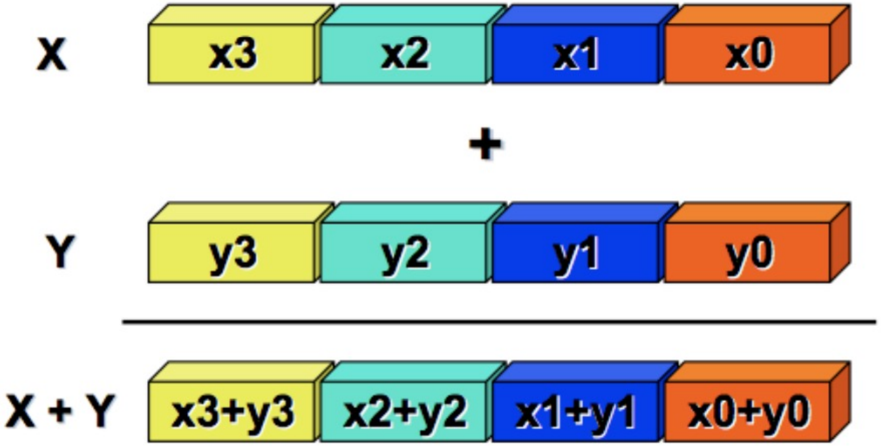
- Corresponds to SIMD architecture



Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

Data-level parallelism

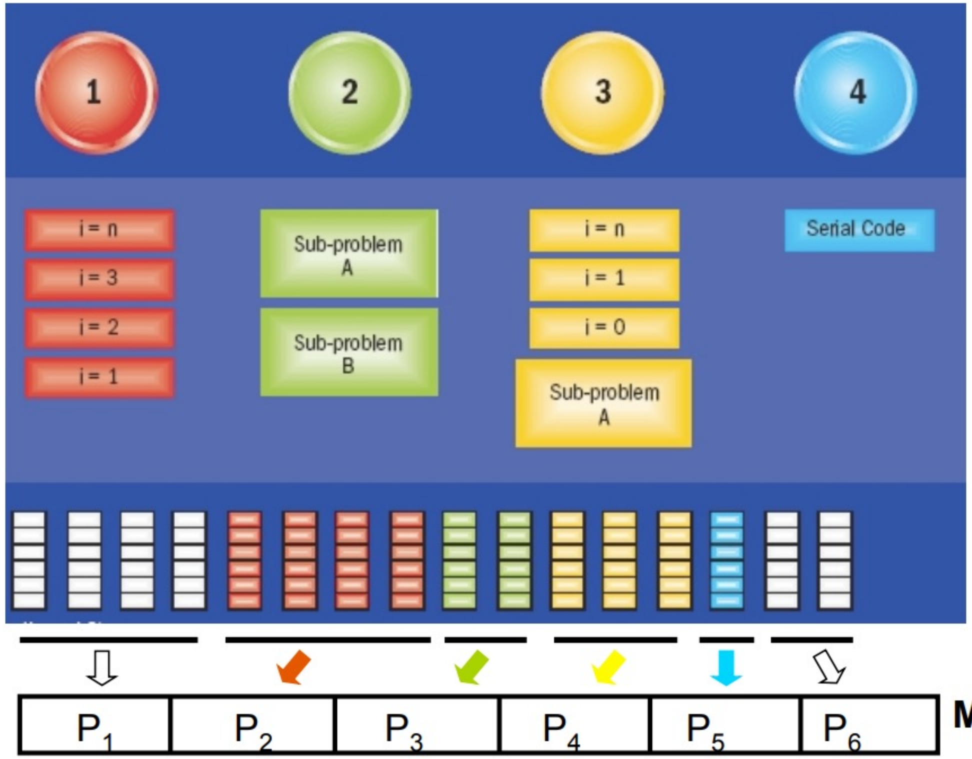
- Corresponds to SIMD architecture
- Example of a sum in parallel:



Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

Parallelism at the thread level

- Corresponds to MIMD architecture



Program partitioned into four threads.

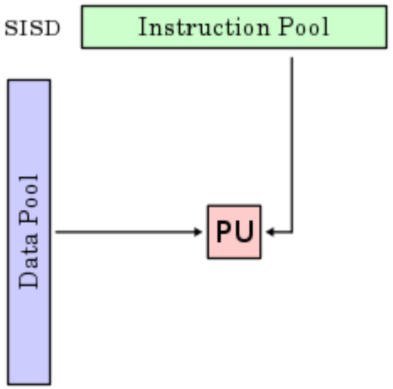
Four threads each executed on separate cores.

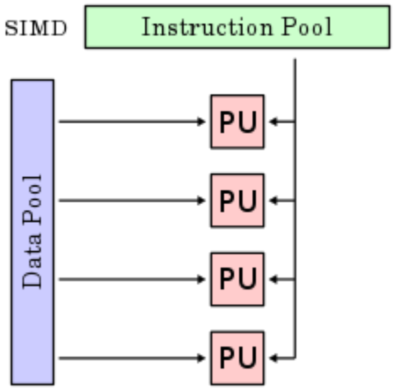
Multicore with 6 cores.

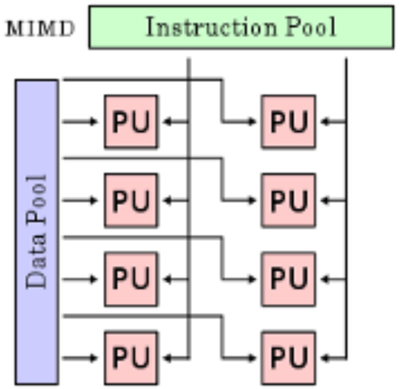
Slide Source: S. Amarasinghe, MIT 6189 IAP 2007

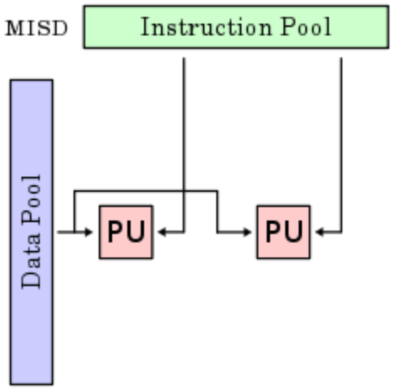
Programming Models

- How to program the different computing models?







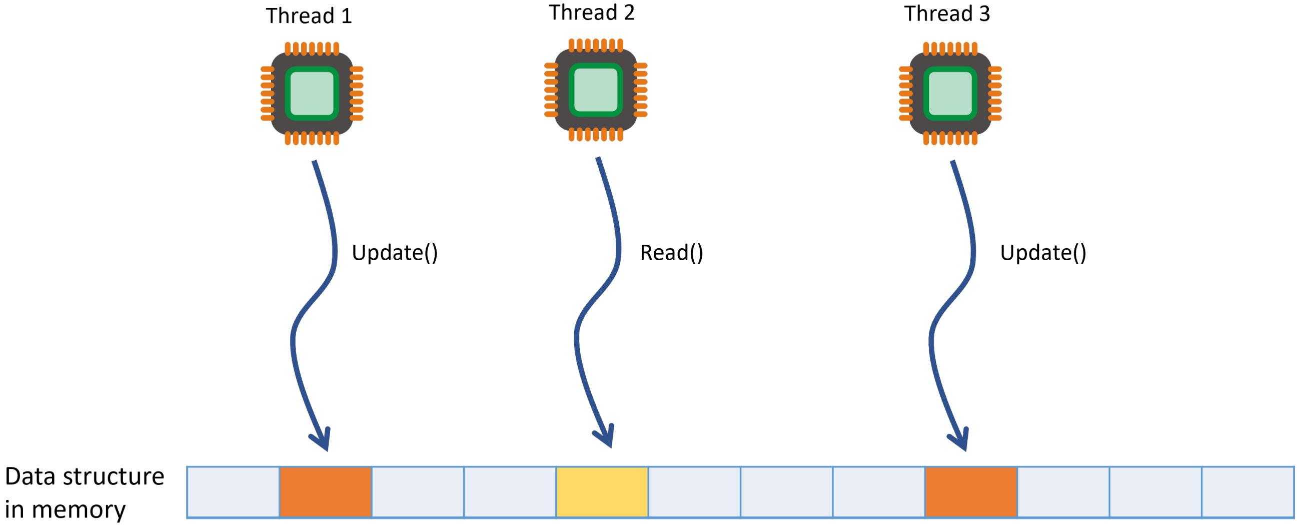


Programming Models

- How to program the different computer models?
- How do we maximize parallelism?
- In practice there are 3 major approaches used in modern processors and accelerators.
  - Multi-threaded programming
  - Multi-threaded programming with multiple data (SIMD)
  - Multi-thread programming with individual data (SIMT)

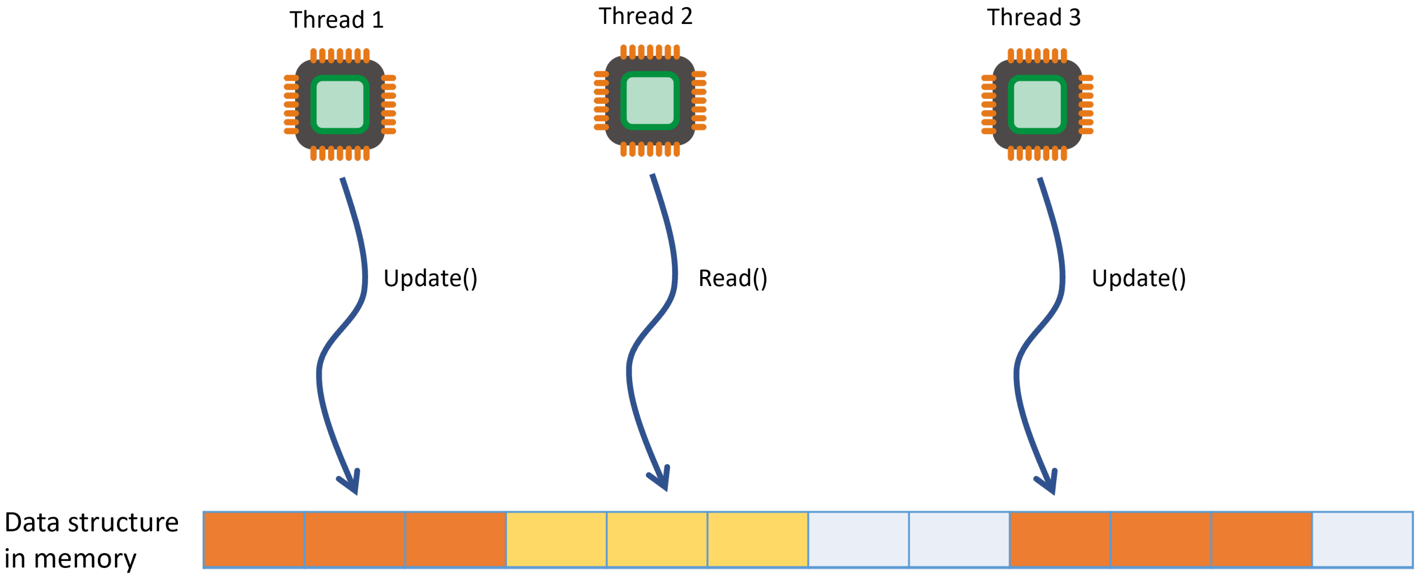
Programming Models: Multi-Threaded

- Multi-threaded parallelism
- Found in modern CPUs
- Multiple threads access individual data from shared data structures in main memory.



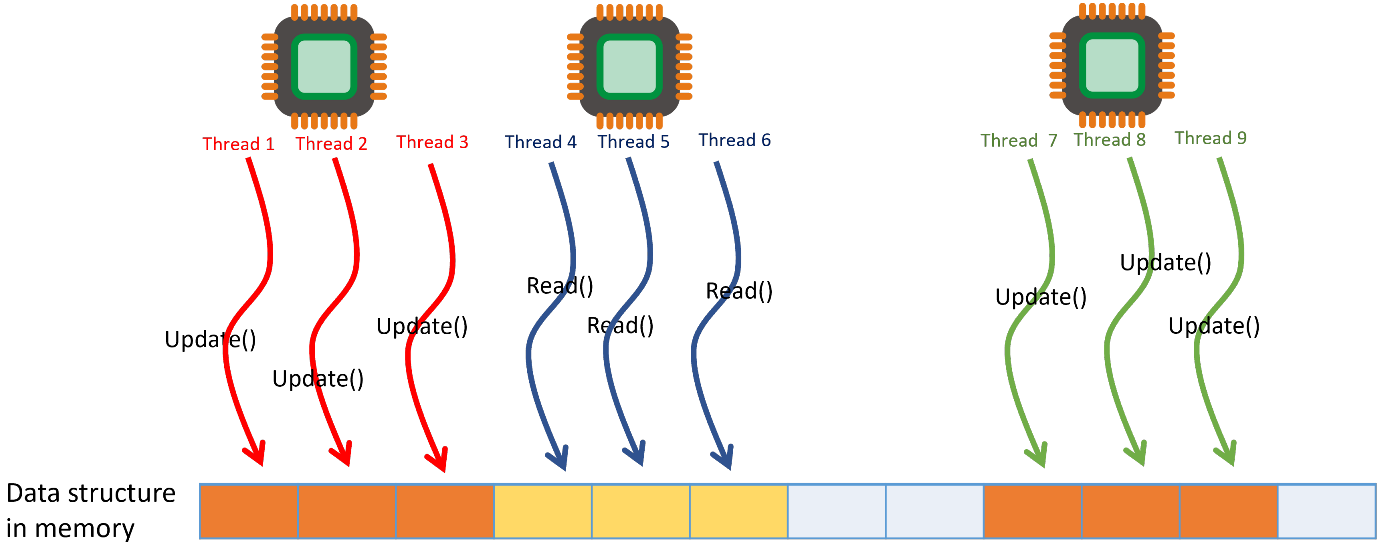
Programming Models: Multi-Threaded SIMD

- Multi-threaded parallelism + data parallelism
- Found in GPUs and some CPUs (e.g. with AVX support)
- Multiple threads access multiple data from shared data structures in main memory.



Programming Models: Multi-Threaded SIMT

- Multi-threaded parallelism in SIMD computing units
- Found in GPUs and AI accelerators
- Multiple threads access individual data by executing the same instruction



Programming Languages and Frameworks

- SISD
  - C++, Java, etc.
- MIMD
  - C++, Java, OpenMP
- SIMD
  - DPC++, C-for-Metal, OpenCL, C++ con extensions
- SIMT
  - OpenCL, CUDA, DPC++

Programming Languages and Frameworks

- CPU multi-core
  - C++, Java, OpenMP, DPC++
- GPU
  - OpenCL, CUDA, DPC++, OpenACC
- FPGA
  - OpenCL, DPC++

References

- S. Amarasinghe, MIT 6189 IAP 2007
- John Cavazos A General Discussion on Parallelism. University of Delaware http://www.cis.udel.edu/~cavazos/cisc879