Multicore Architecture Programming – Introduction

Olivier Aumage

Inria & LaBRI lab.

olivier.aumage @ inria.fr

2024 – 2025

Introduction

Context

Rapidly evolving processor technologies

- **Density**
  - 2024
    - Intel Emerald Rapids architecture: up to 64 cores [link] (Dec. 2023)
      - 2024: Granite Rapids (up to 128 P-cores: HPC) / Sierra Forrest (up to 288 E-cores: Cloud Computing)
    - AMD EPYC Turin / Zen 5 architecture: up to 192 cores [link]
- **Complexity & Optimization trade-off**
  - Bandwidth
  - Latency
  - Energy consumption & power envelope
  - Cost
- **Heterogeneity**
  - Specialized instruction sets
  - Accelerators & specialized processors

Objectives

Purpose of the course

- **Understand key design principles of modern processors**
- **Learn and combine techniques to write efficient programs for such processors**
- **Get ready to adapt in a rapidly evolving landscape**

Organisation

Speakers

- **Olivier Aumage**
  - General purpose multicore programming
  - Single Instruction Multiple Data (SIMD) programming
  - Task parallelism

- **Amina Guermouche**
  - Energy & power consumption in multicore processors
  - Accelerator architectures programming

Organisation

Evaluation

- **Assignment + Oral defense**
    - Apply optimization techniques on sample programs

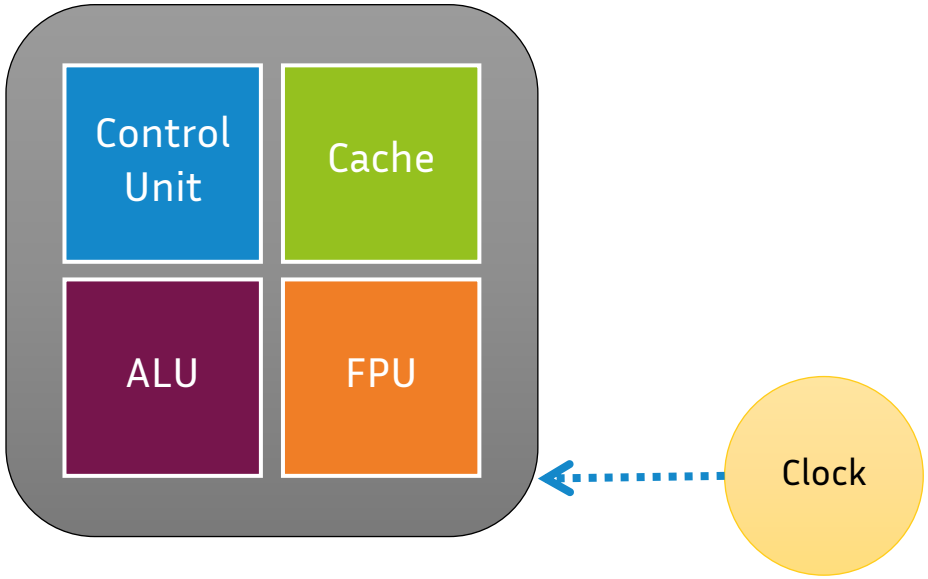
Basic Architectural Notions

Refresh

Common Architectural Principles of Microprocessors

Schematic Design

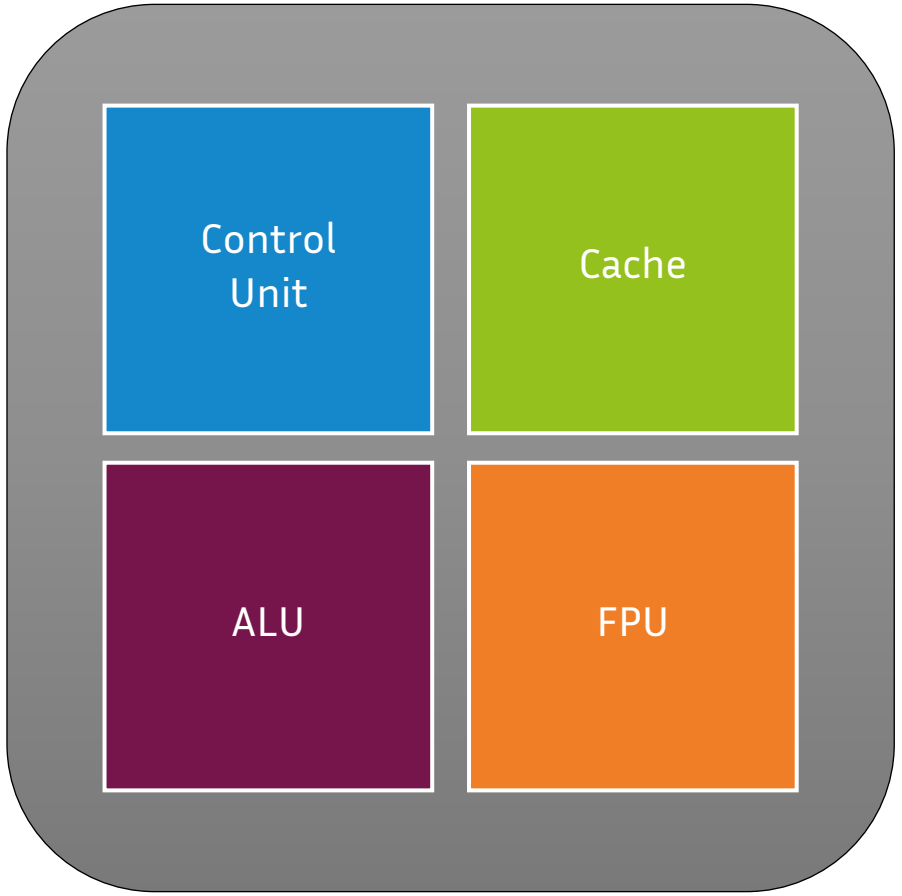
- **Die areas**
  - Control unit
    - Register file
  - Arithmetic & logic unit (ALU)
    - Integer operations
  - Floating point unit (FPU)
  - Cache (s)
    - Purposes
    - Levels



Common Architectural Principles of Microprocessors

Schematic Design

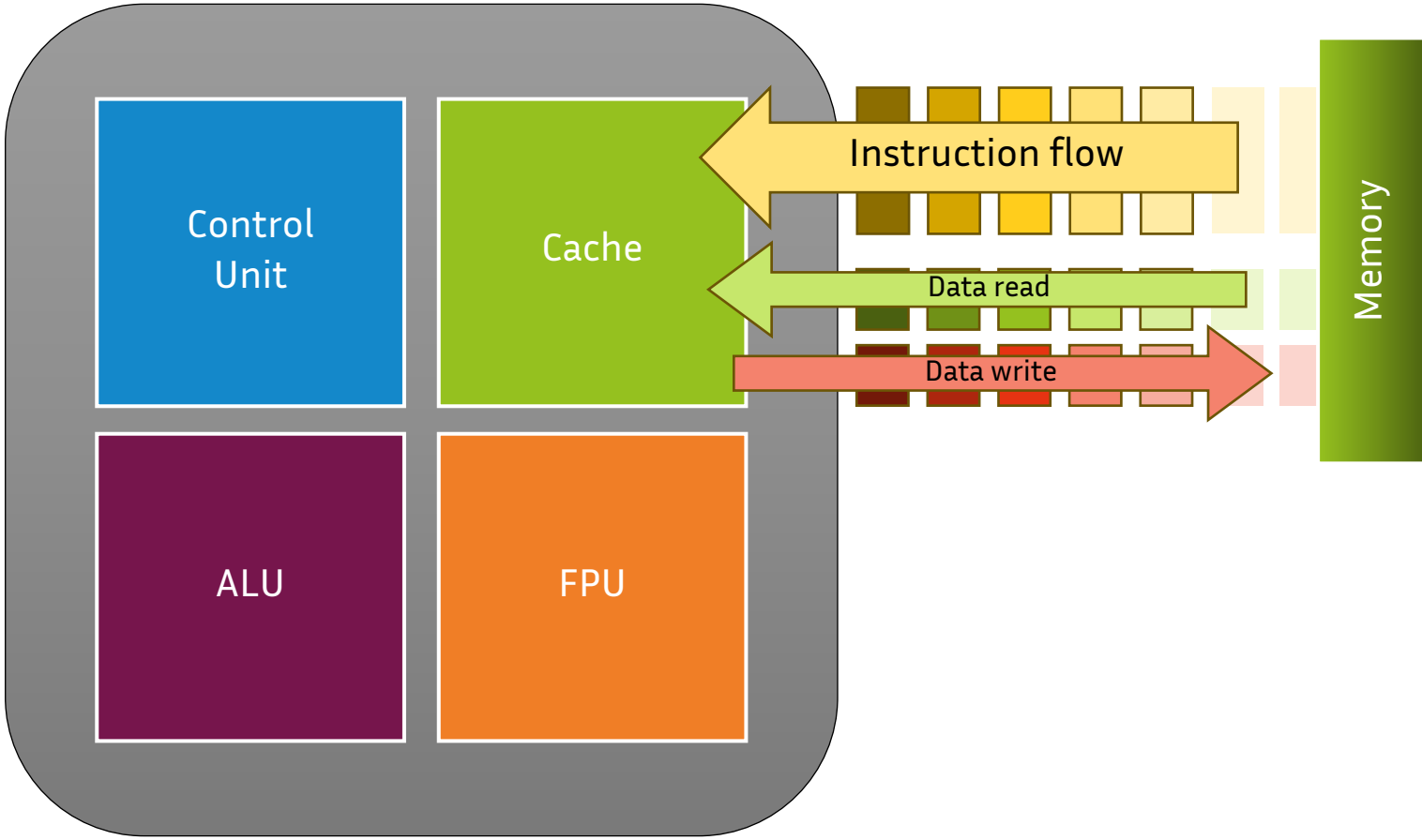
- **Die areas**
  - Control unit
    - Register file
  - Execution units
    - Arithmetic & logic unit (ALU)
    - Floating point unit (FPU)
  - Cache (s)
    - Purposes
    - Levels
- **Instruction flow**
  - Thread



Common Architectural Principles of Microprocessors

Schematic Design

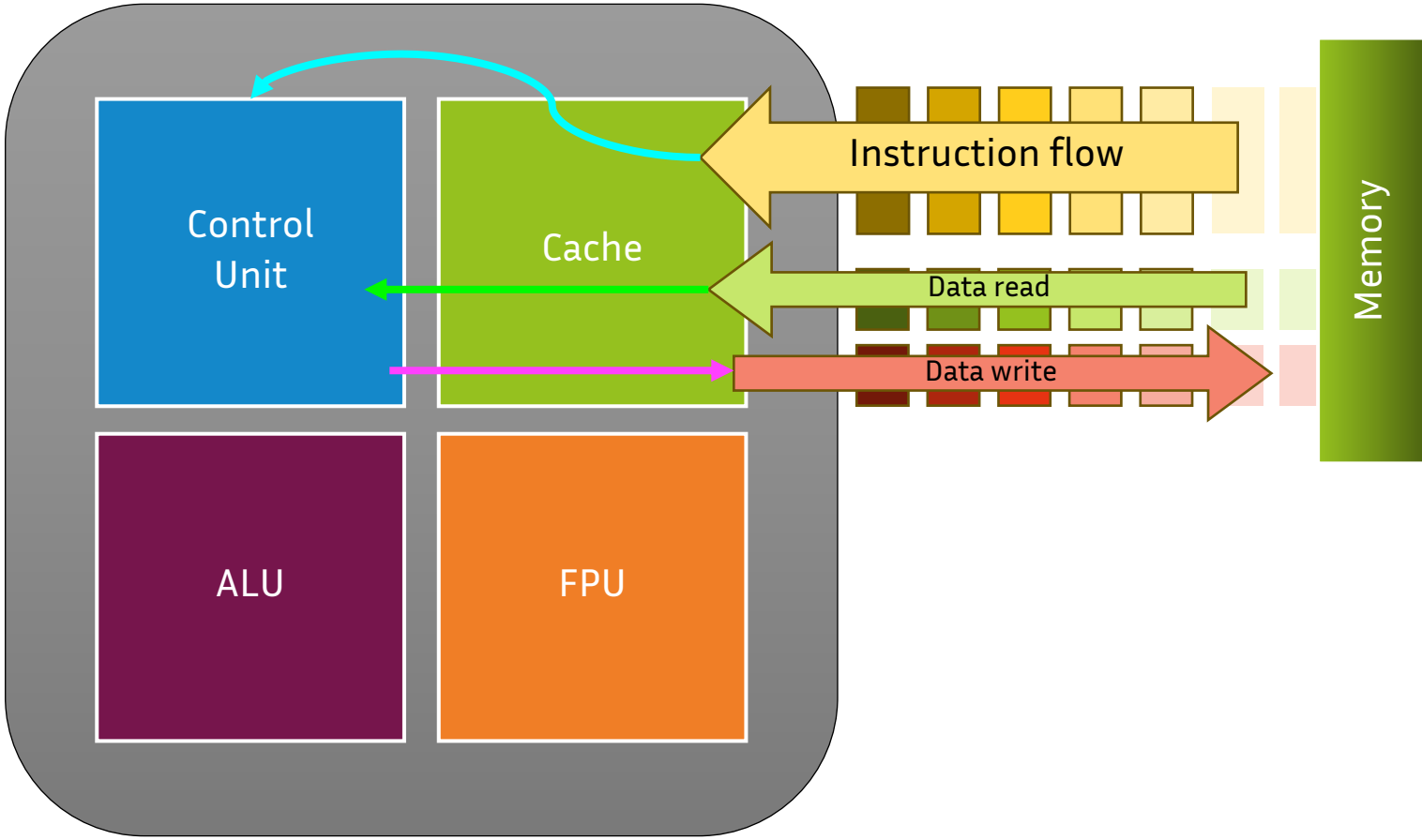
- **Die areas**
  - Control unit
    - Register file
  - Execution units
    - Arithmetic & logic unit (ALU)
    - Floating point unit (FPU)
  - Cache (s)
    - Purposes
    - Levels
- **Instruction flow**
  - Thread



Common Architectural Principles of Microprocessors

Schematic Design

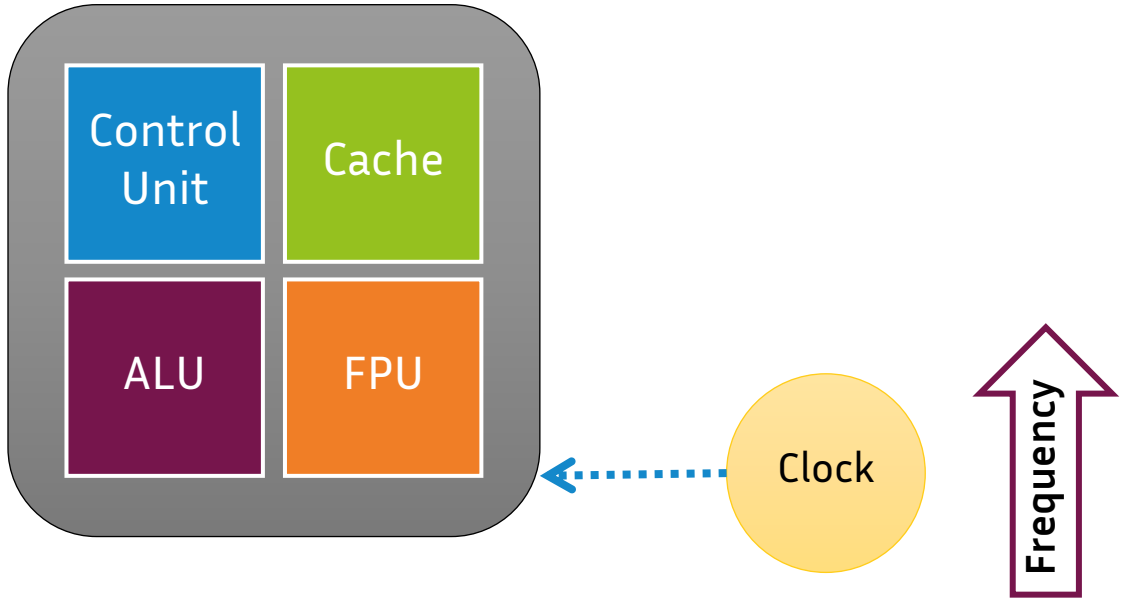
- **Die areas**
  - Control unit
    - Register file
  - Execution units
    - Arithmetic & logic unit (ALU)
    - Floating point unit (FPU)
  - Cache (s)
    - Purposes
    - Levels
- **Instruction flow**
  - Thread



Improving Performances

Computing faster?

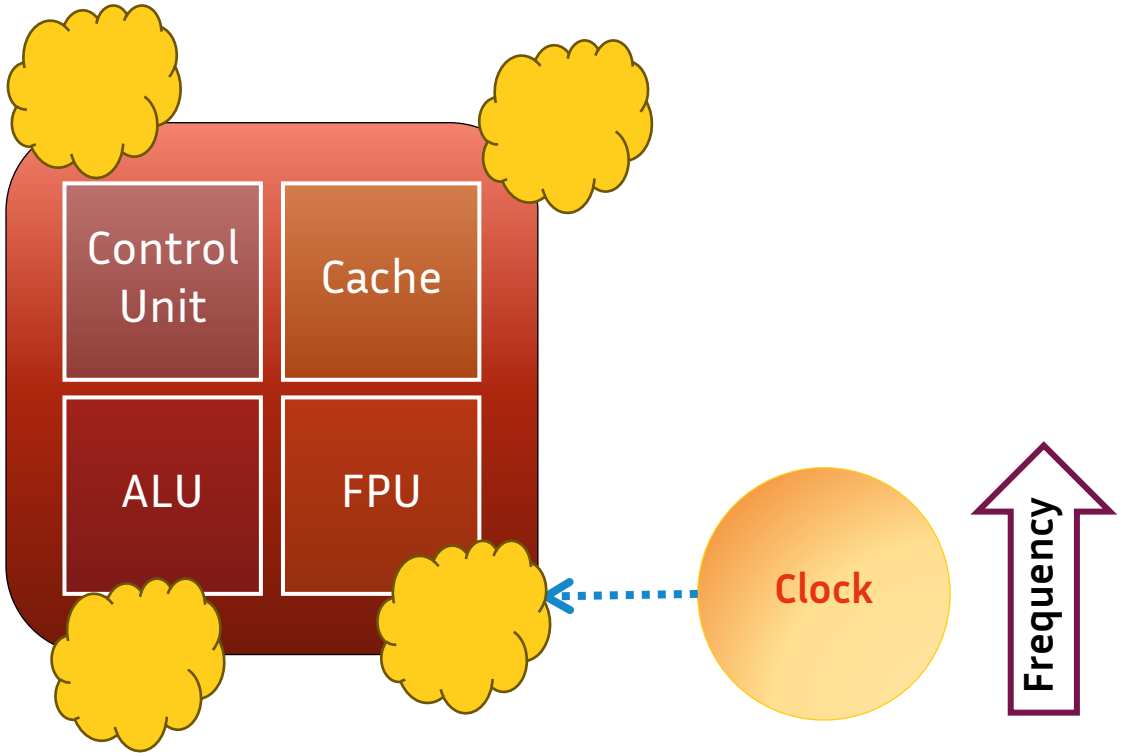
- **Increase frequency?**



Frequency Wall

Physical limits

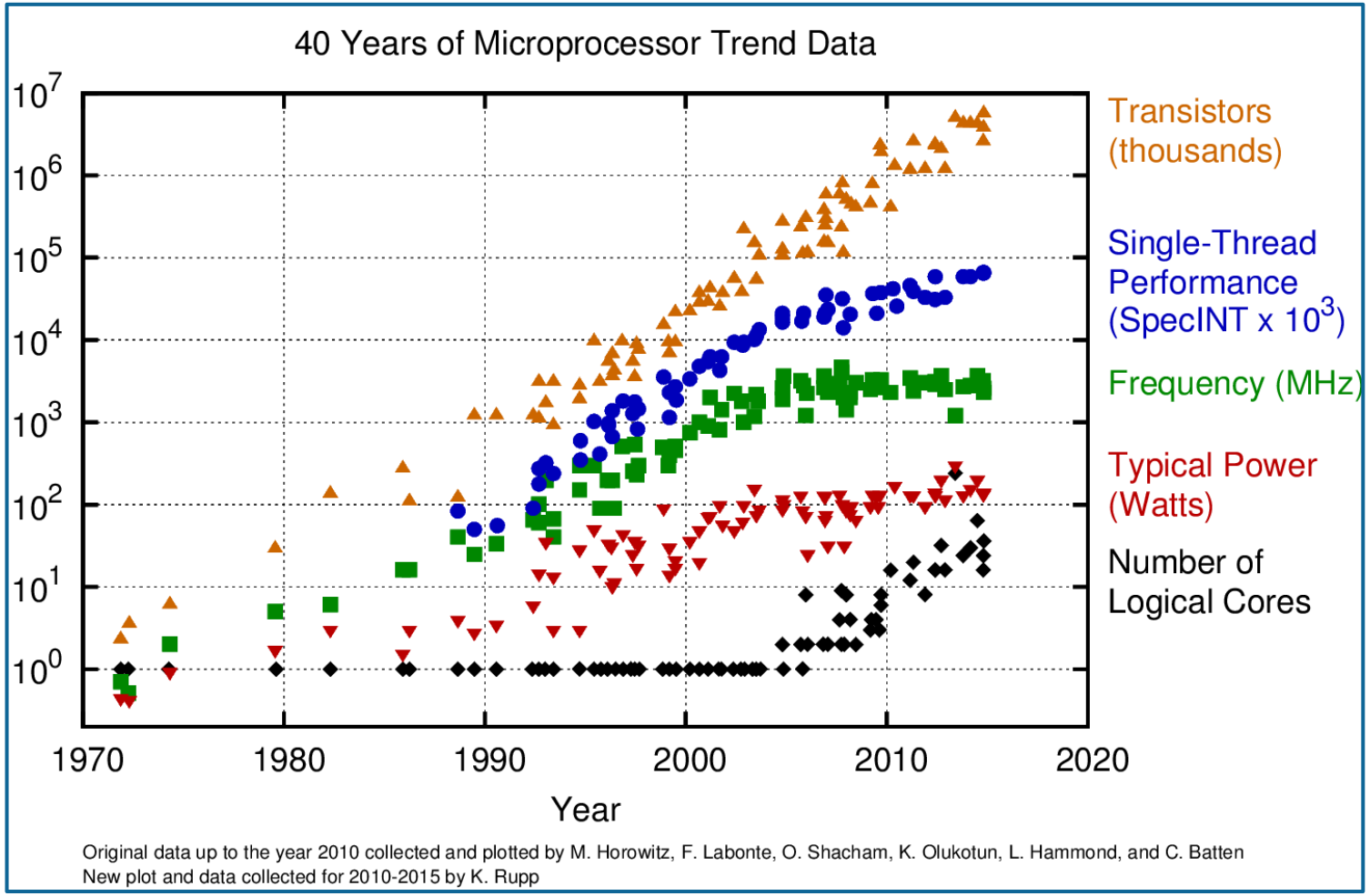
- **Thermal density**
- **Processor cooling**



Heat Dissipation Barrier

Causes & Consequences

- **Processor frequency evolution**

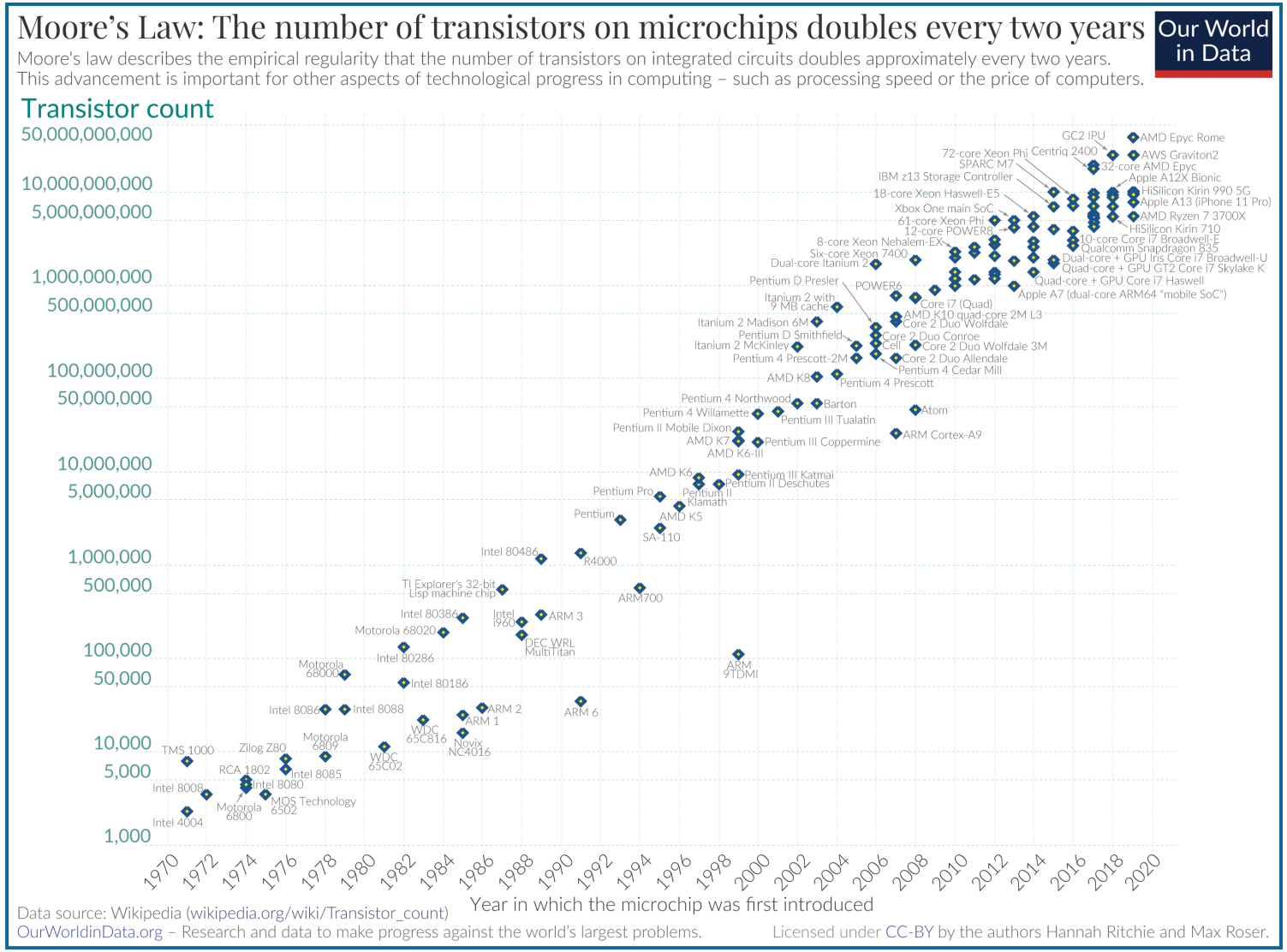


Source: K. Rupp [link]

Moore’s Law

Gordon Moore, Intel

The number of transistors in an IC doubles every two years



Source: Max Roser, Hannah Ritchie, 2020

Mo

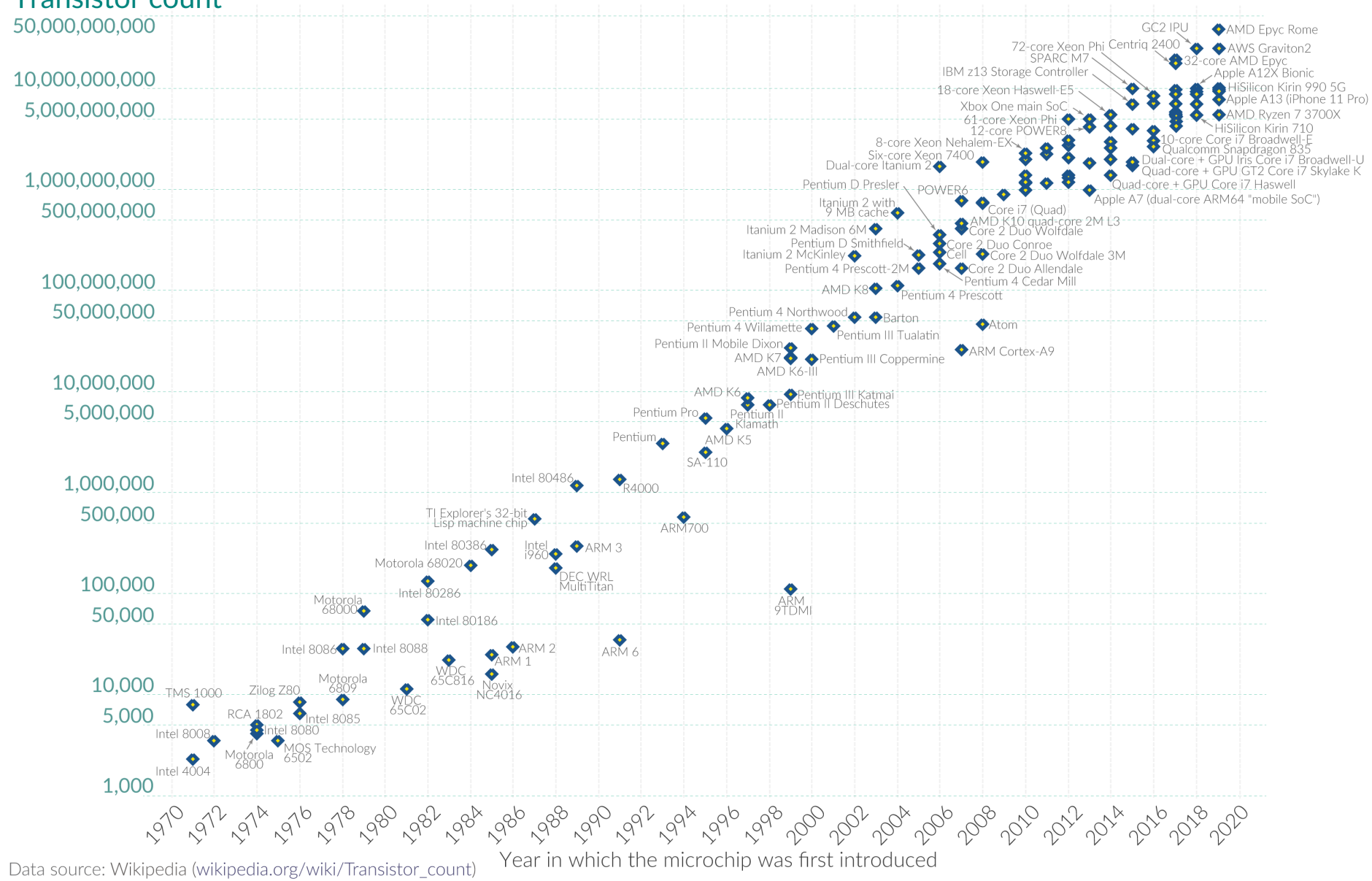
Gordon

The nu
in an I
years

Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World in Data



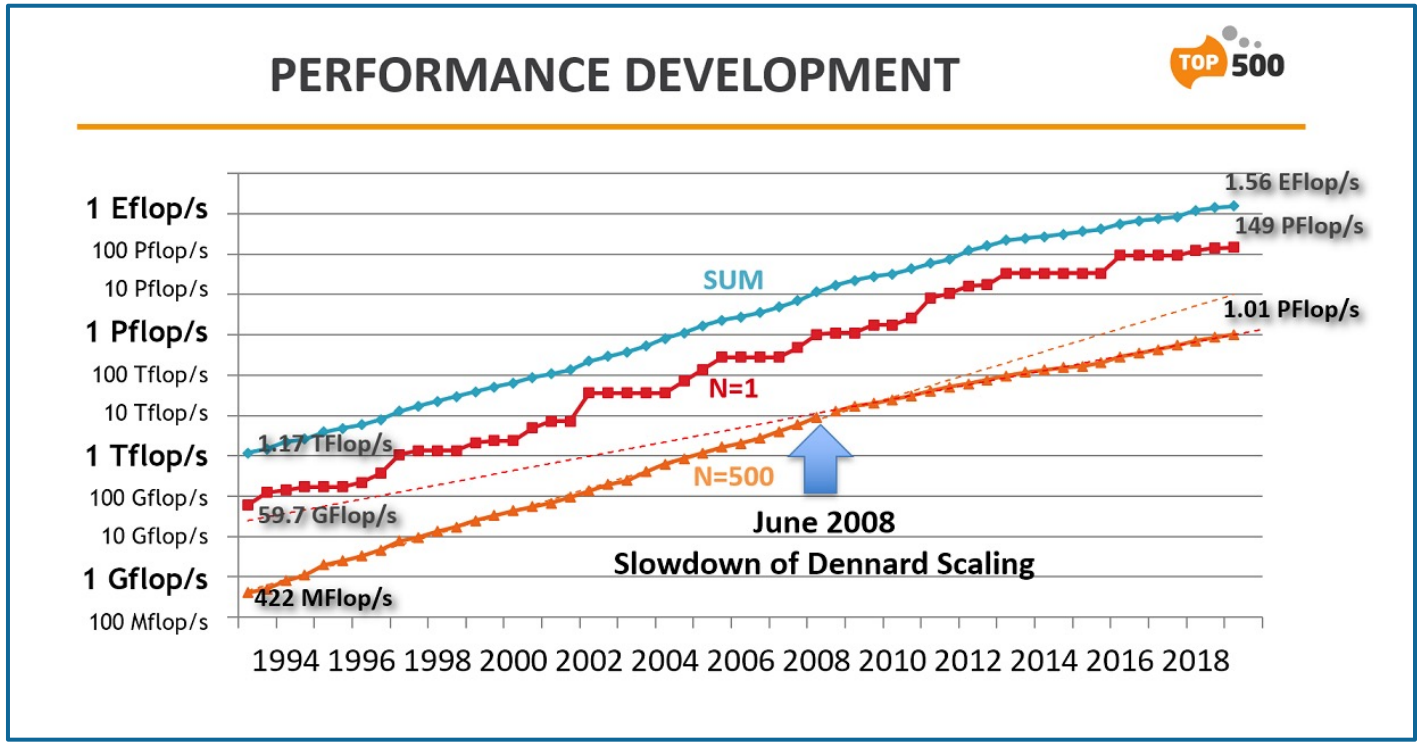
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.

Source: Max Roser, Hannah Ritchie, 2020

Dennard Scaling

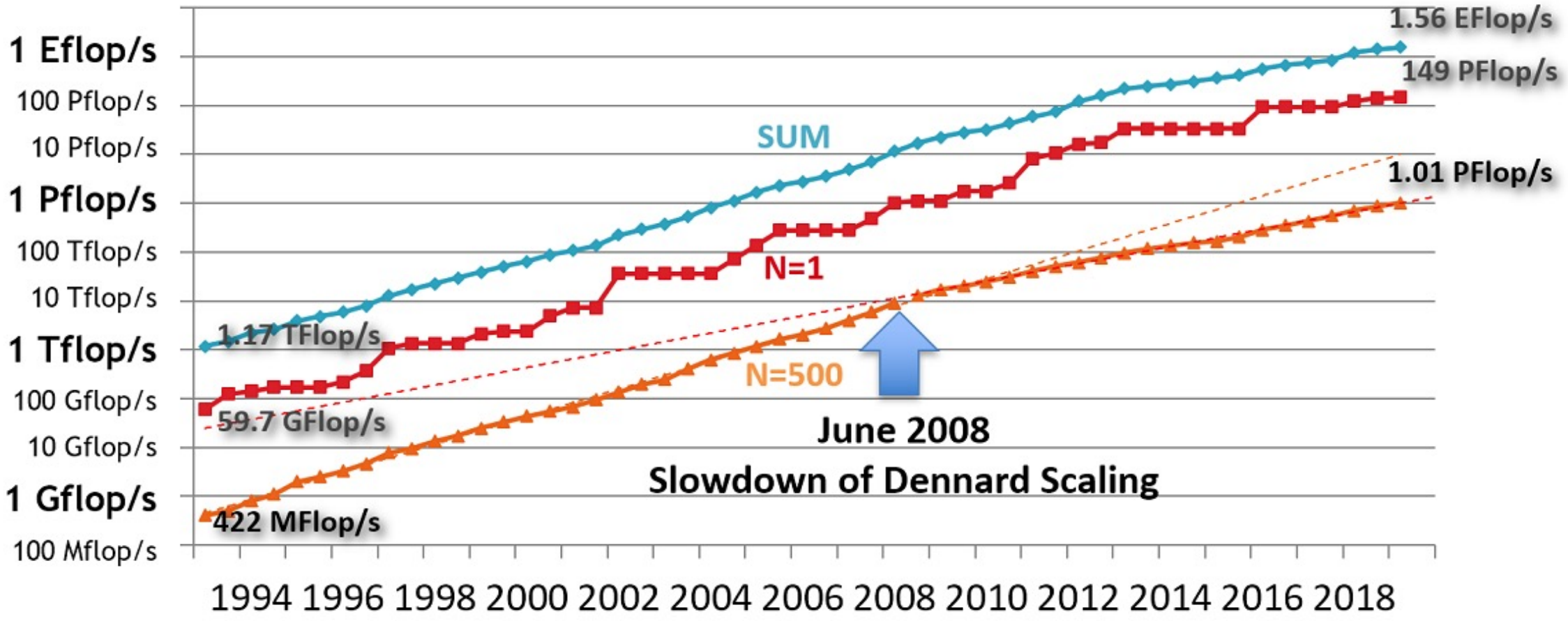
Robert H. Dennard, IBM

- **As transistors gets smaller, their power density stays constant**
  - Power use in proportion with area
- **Transition around 2006 – 2008**



Source: M. Feldman, The Next Platform [link]

PERFORMANCE DEVELOPMENT



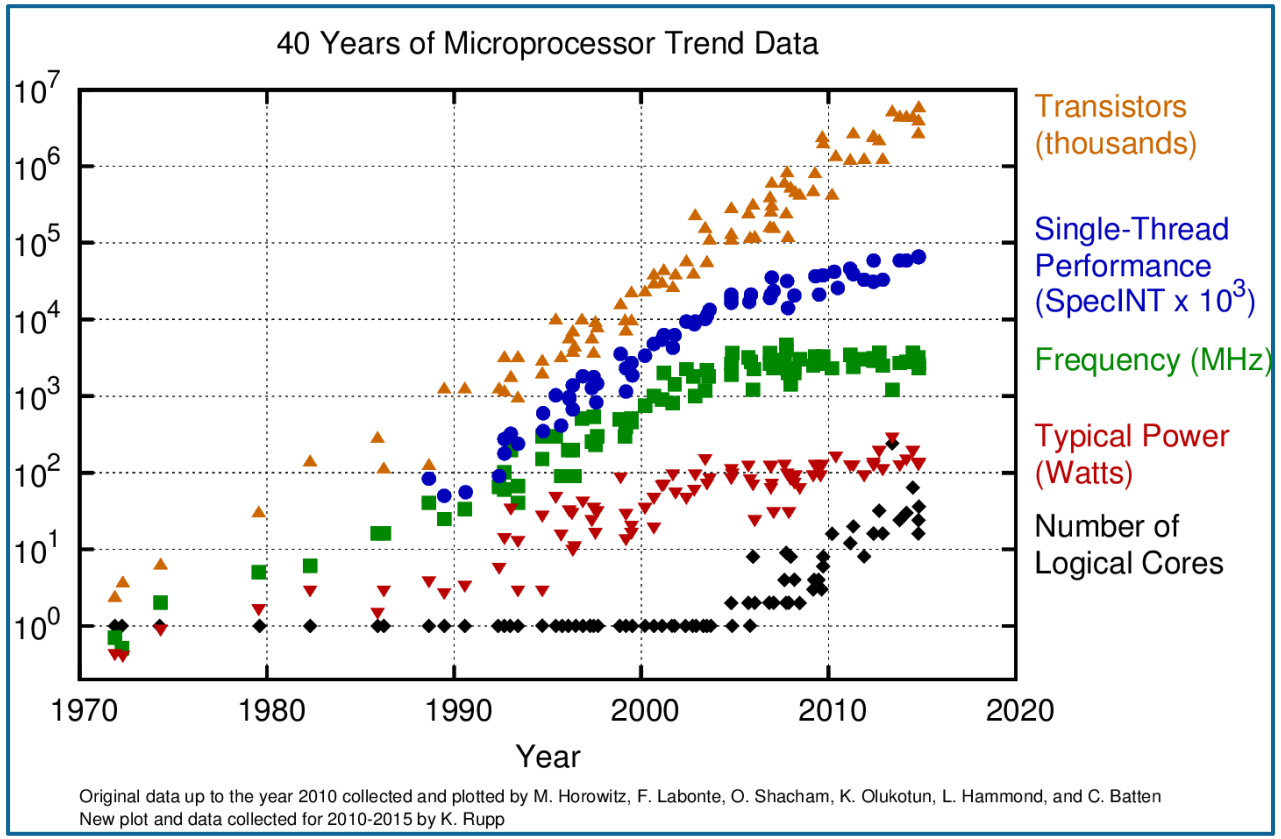
Source: M. Feldman, The Next Platform [link]

Heat Dissipation Barrier

Causes & Consequences

- Processor frequency evolution
- Moore's Law
- Dennard Scaling

Improve performances by other means



Source: K. Rupp [link]

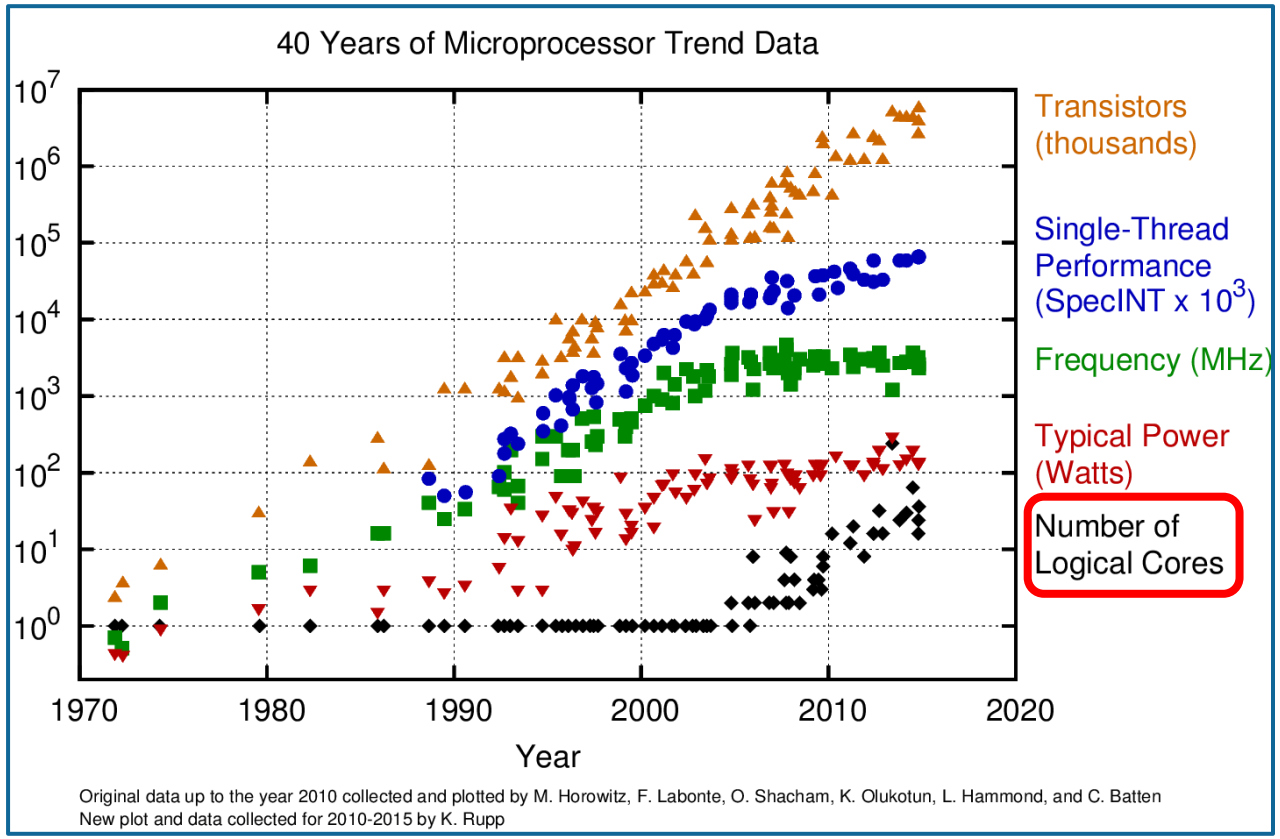
Heat Dissipation Barrier

Causes & Consequences

- Processor frequency evolution
- Moore's Law
- Dennard Scaling

Improve performances by other means

- Doing more by unit of time
- **Parallelism**

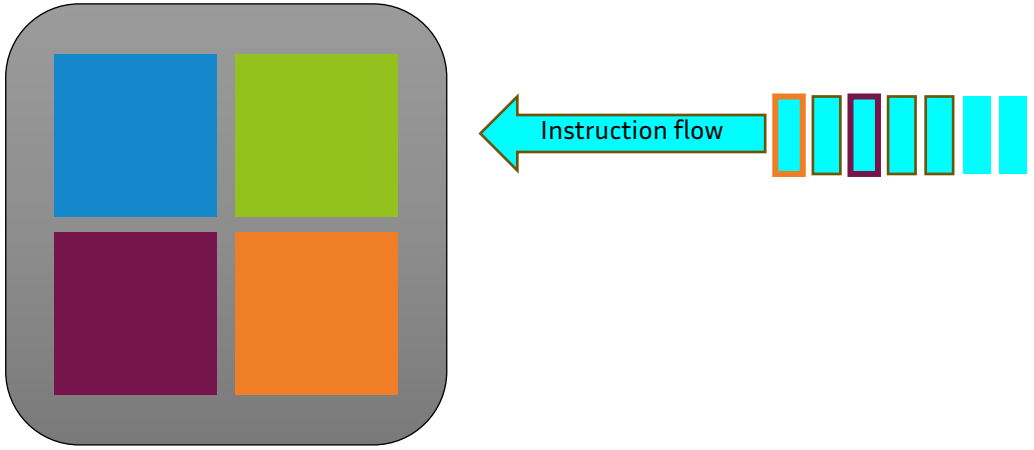


Source: K. Rupp [link]

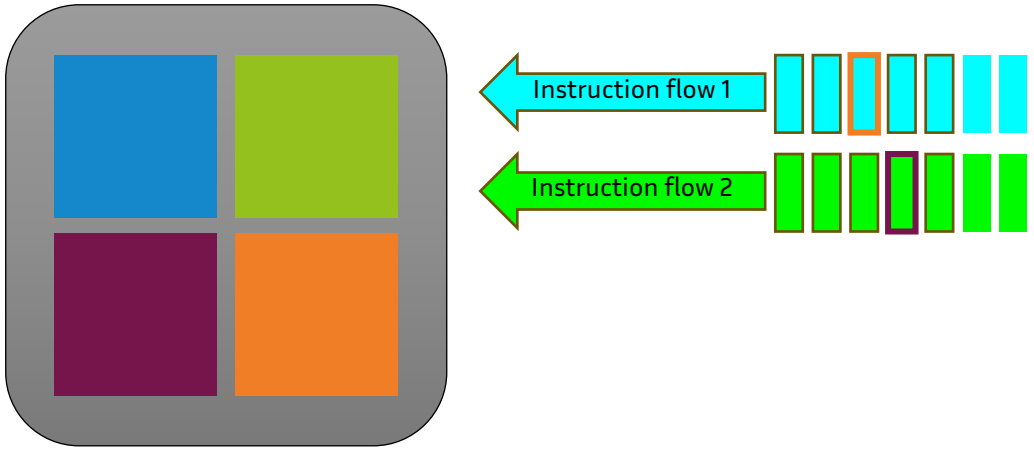
Doing More by Unit of Time

Hardware parallelism

- **Instruction-level parallelism (ILP)**
  - Parallelism within a single instruction flow



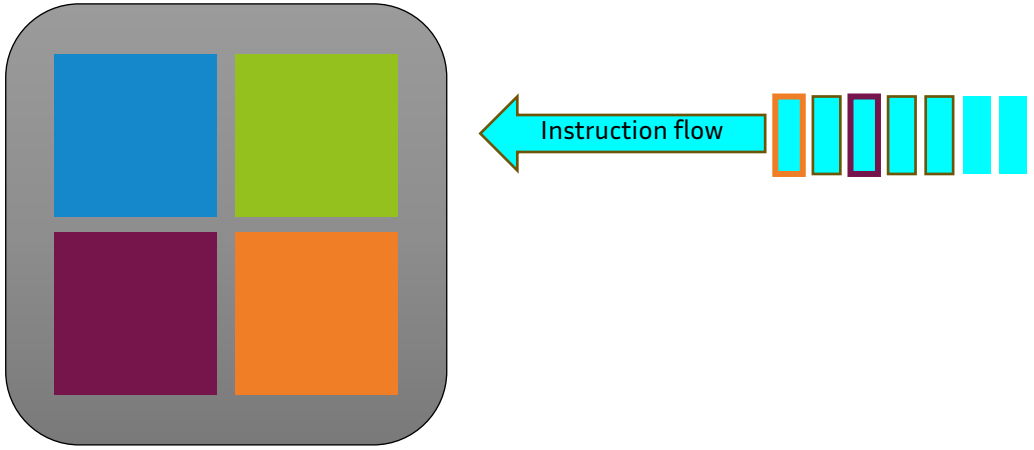
- **Thread-level parallelism (TLP)**
  - Parallelism across multiple instruction flows



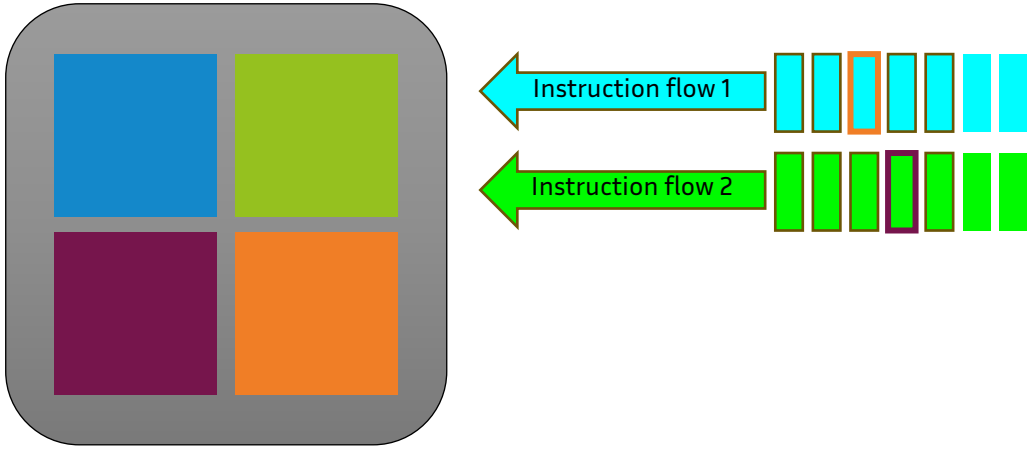
Doing More by Unit of Time

Hardware parallelism

- **Instruction-level parallelism (ILP)**
  - Parallelism within a single instruction flow



- **Thread-level parallelism (TLP)**
  - Parallelism across multiple instruction flows



Instruction-Level Parallelism

Extract parallelism within a single flow of instructions

- **Overlap the processing of successive instructions**
  - Pipelining
- **Process several instructions at a given time**
  - Overlapping / offload
  - Superscalar processing
- **Apply an instruction on multiple data**
  - SIMD instruction sets

Instruction-Level Parallelism

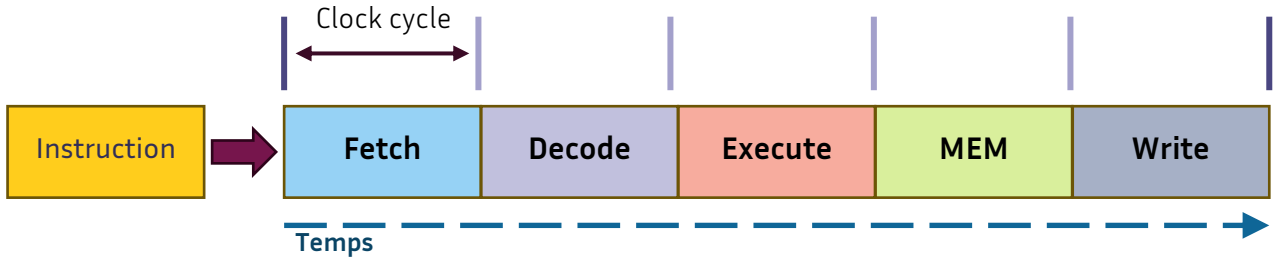
Extract parallelism within a single flow of instructions

- **Overlap the processing of successive instructions**
  - Pipelining
- **Process several instructions at a given time**
  - Overlapping / offload
  - Superscalar processing
- **Apply an instruction on multiple data**
  - SIMD instruction sets

Pipelining

Overlap Instruction Processing Steps

- **Multiple instruction execution steps**
- **1 step >= 1 clock cycle**

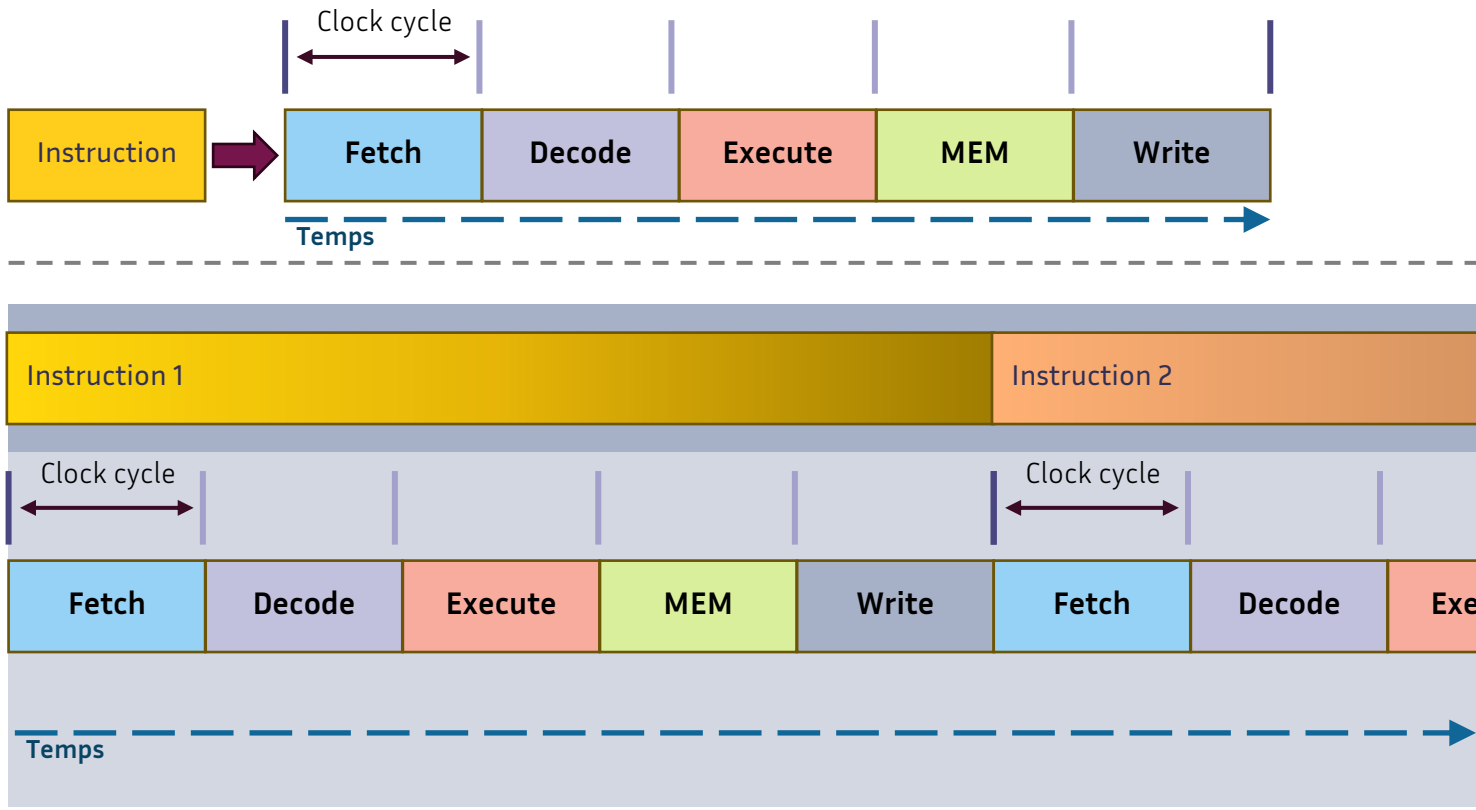


Typical RISC instruction processing sequence

Pipelining

Overlap Instruction Processing Steps

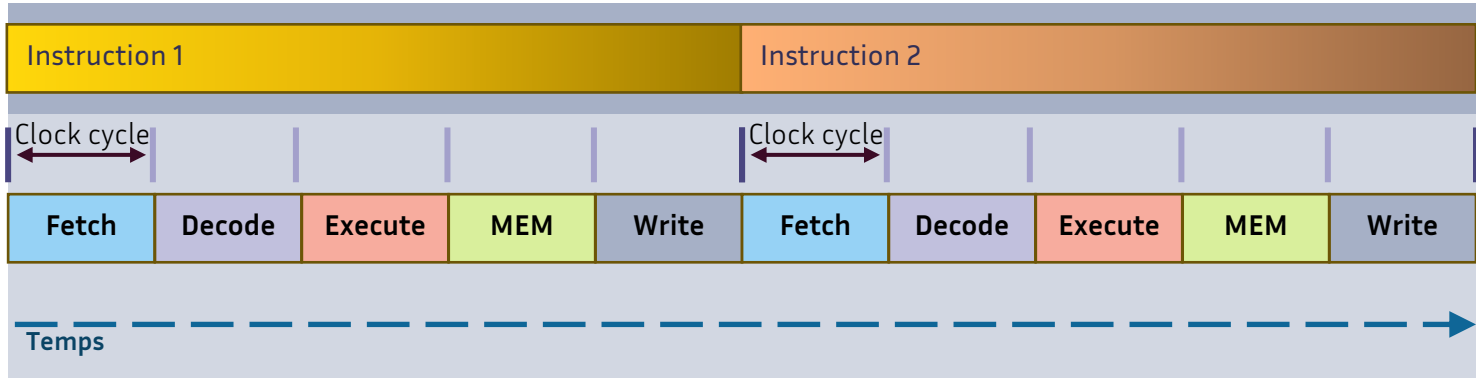
- **Multiple instruction execution steps**
- **1 step >= 1 clock cycle**



Pipelining

Overlap Instruction Processing Steps

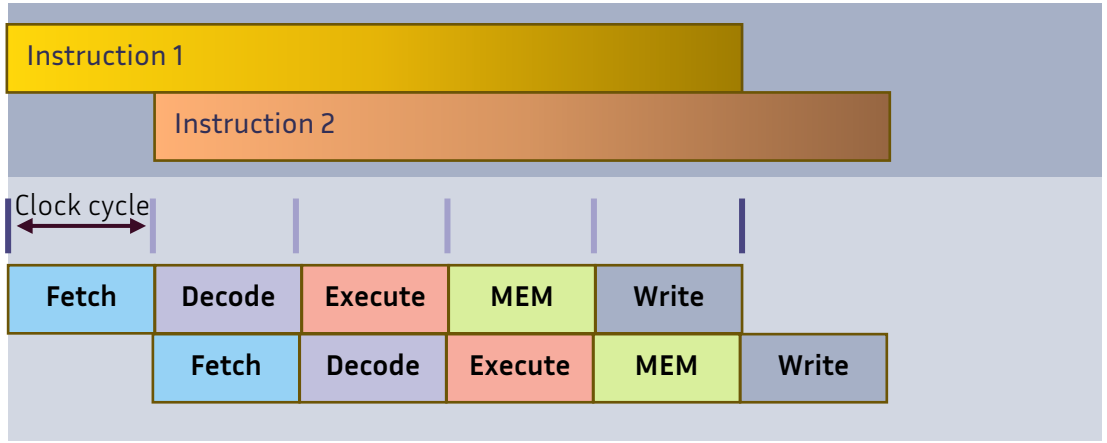
- **Multiple instruction execution steps**
- **1 step >= 1 clock cycle**



Pipelining

Overlap Instruction Processing Steps

- **Multiple instruction execution steps**
- **1 step >= 1 clock cycle**

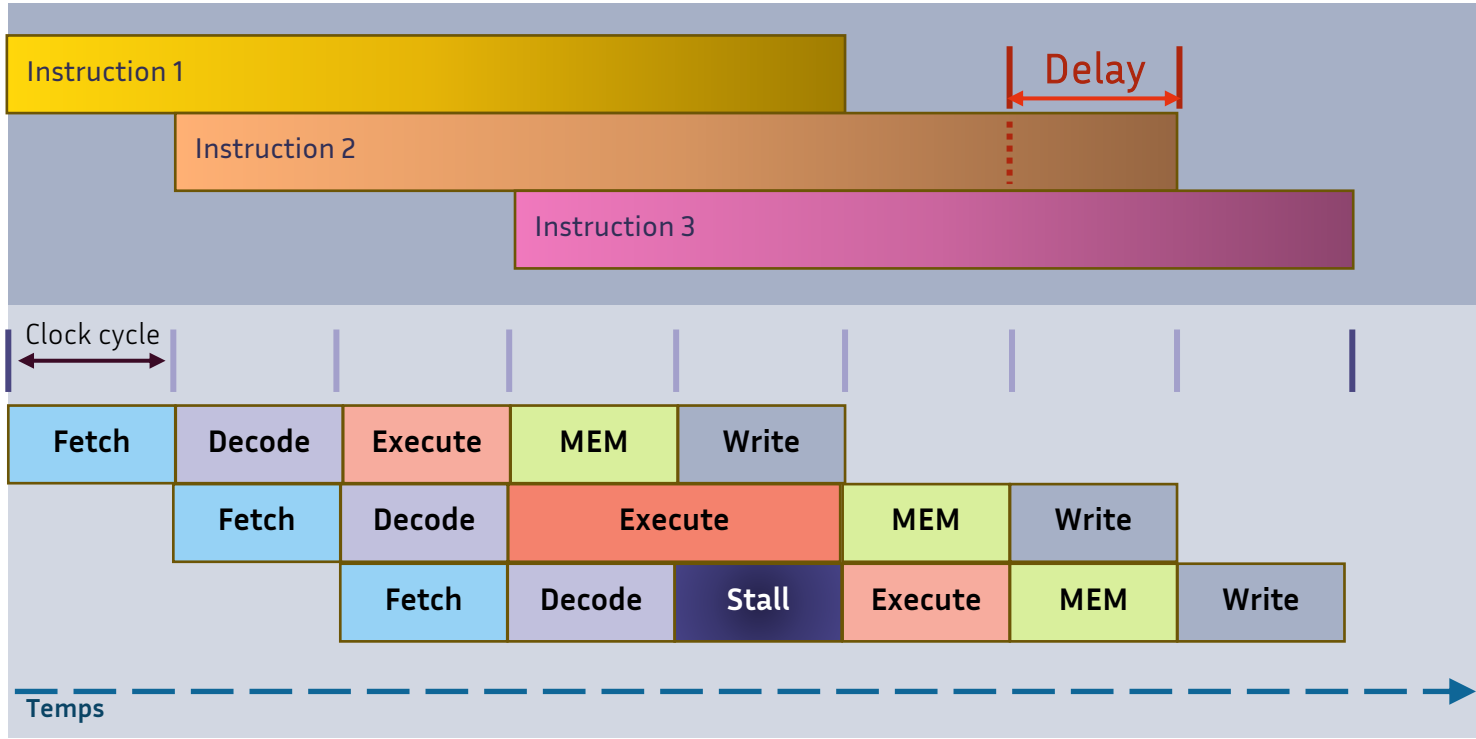


- **Pipelining**
  - Process distinct steps of successive instructions in parallel
- **Improve Instruction per Cycle rate (IPC)**
  - Ideally, 1 instruction per cycle
- **Cost**
  - Additional circuitry to manage multiple steps in parallel

Pipelining

Overlap Instruction Processing Steps

- **Limitations**
  - Pipeline stalls (bubbles)
- **Causes**
  - Long instructions
  - Memory accesses
  - Data dependencies



Instruction-Level Parallelism

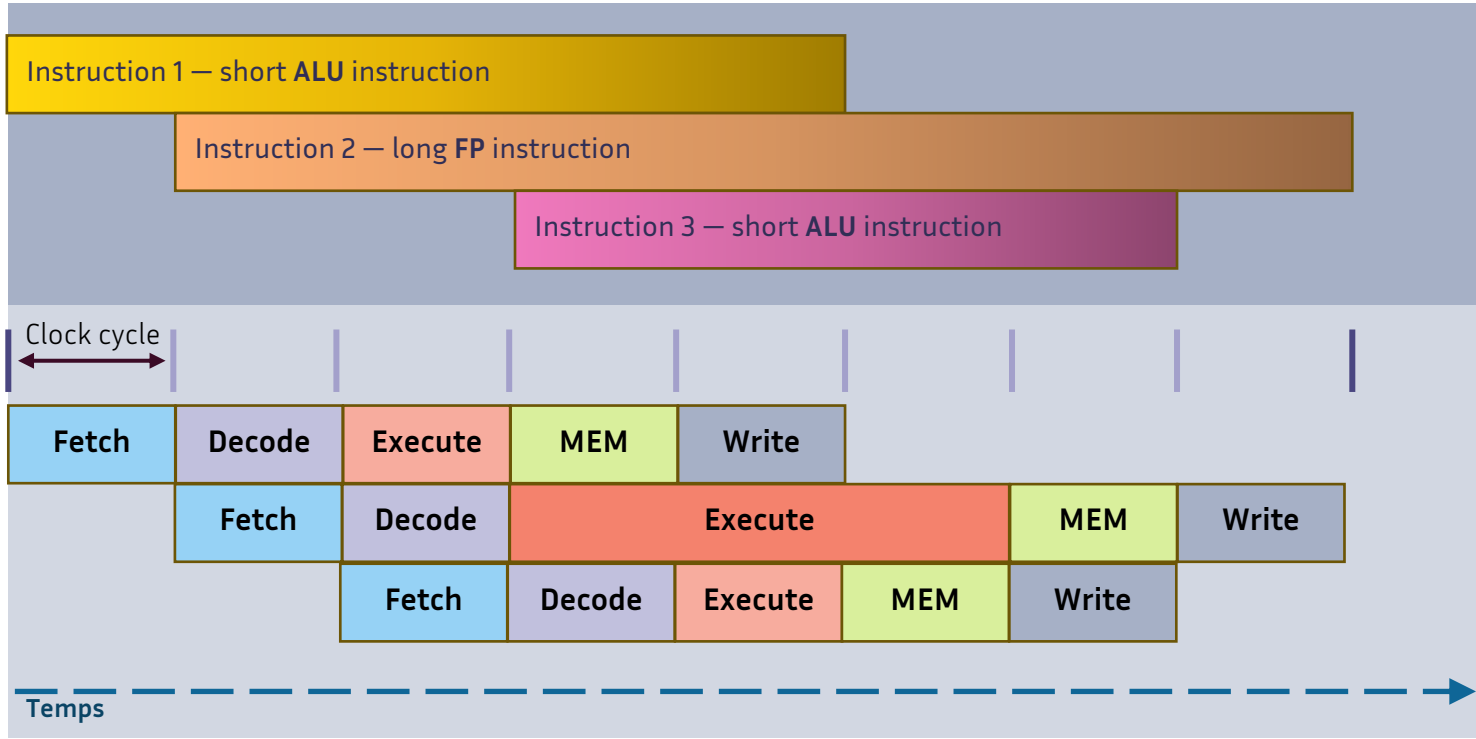
Extract parallelism within a single flow of instructions

- **Overlap the processing of successive instructions**
  - Pipelining
- **Process several instructions at a given time**
  - Overlapping / offload
  - Superscalar processing
- **Apply an instruction on multiple data**
  - SIMD instruction sets

Overlapping / offload

Overlap instructions processing

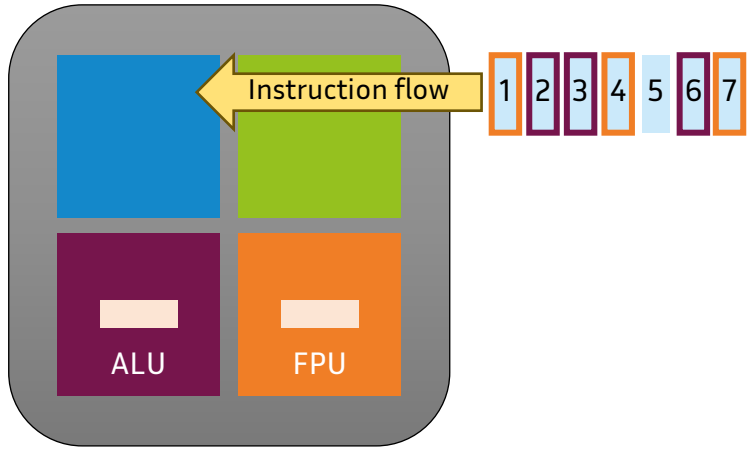
- **Heterogeneous instructions**
  - Using different arithmetic units

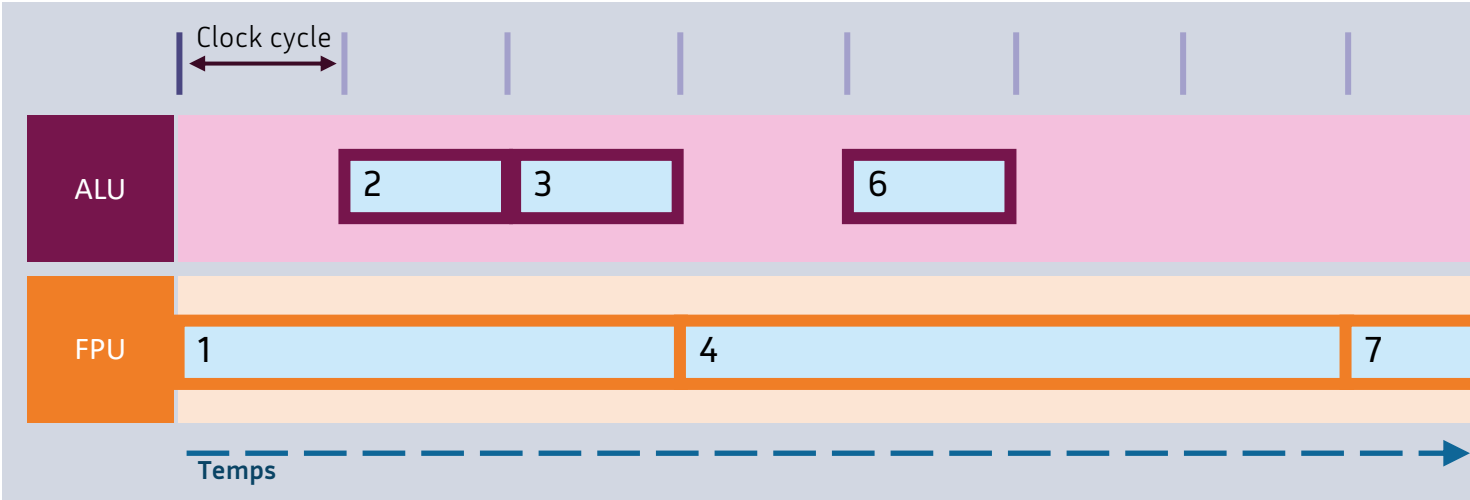


Overlapping / offload

Overlap instructions processing

- **Heterogeneous instructions**
  - Using different arithmetic units

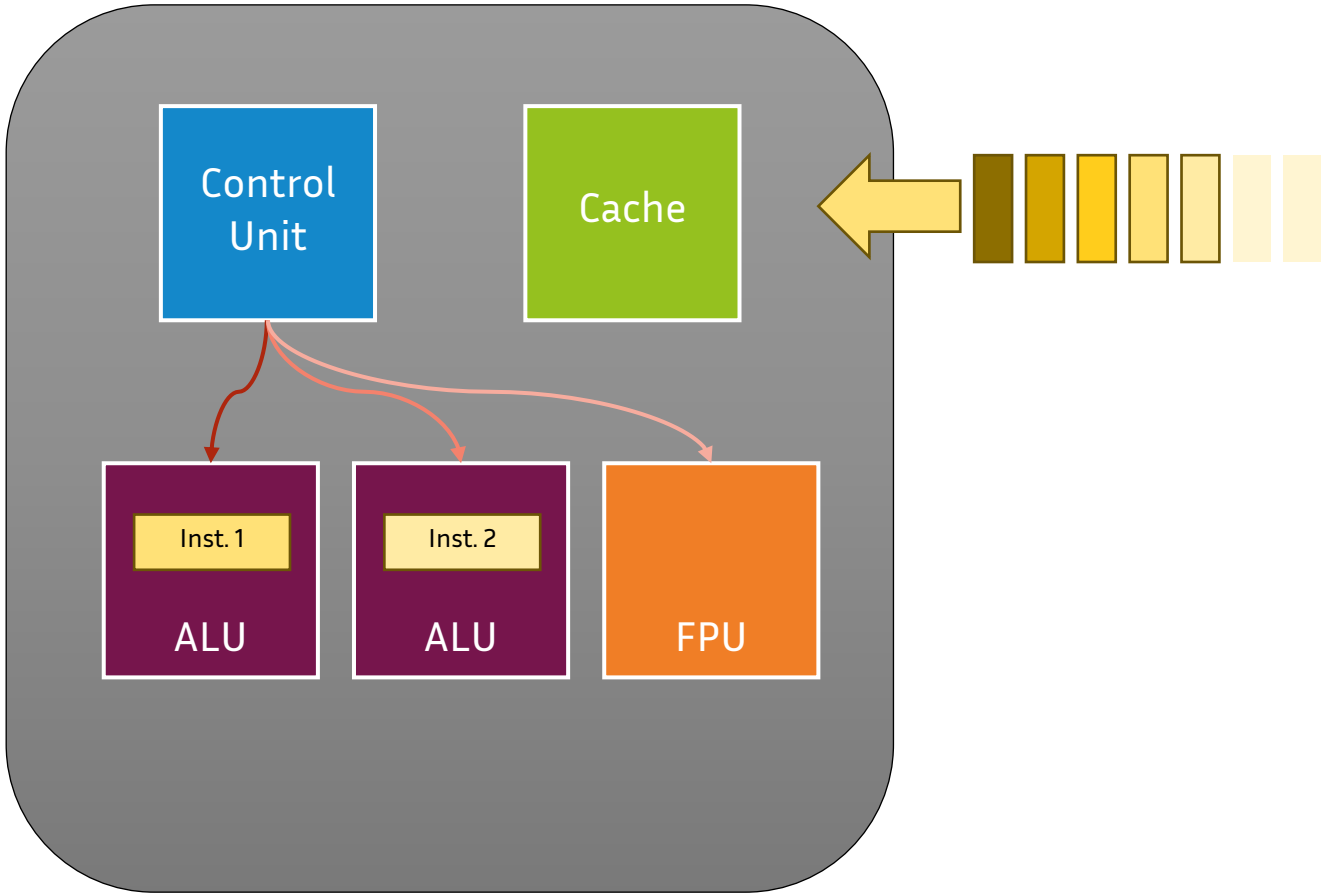




Superscalar Processors

Overlap instructions processing

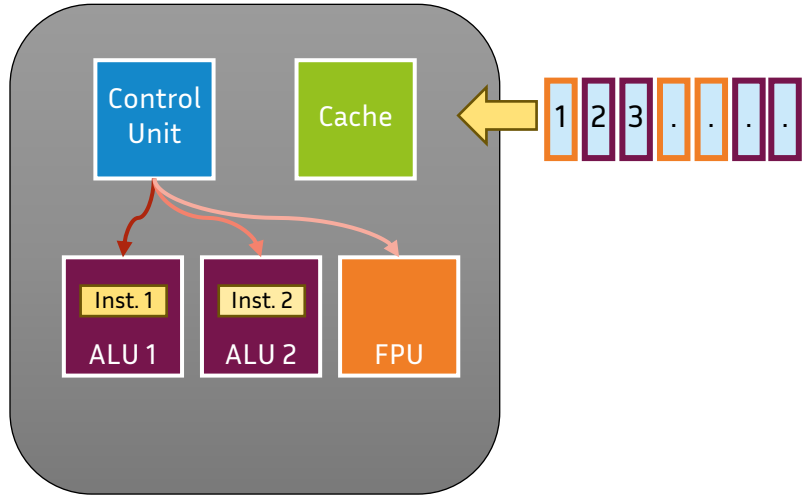
- **Homogeneous instructions**
  - Multiple identical arithmetic units

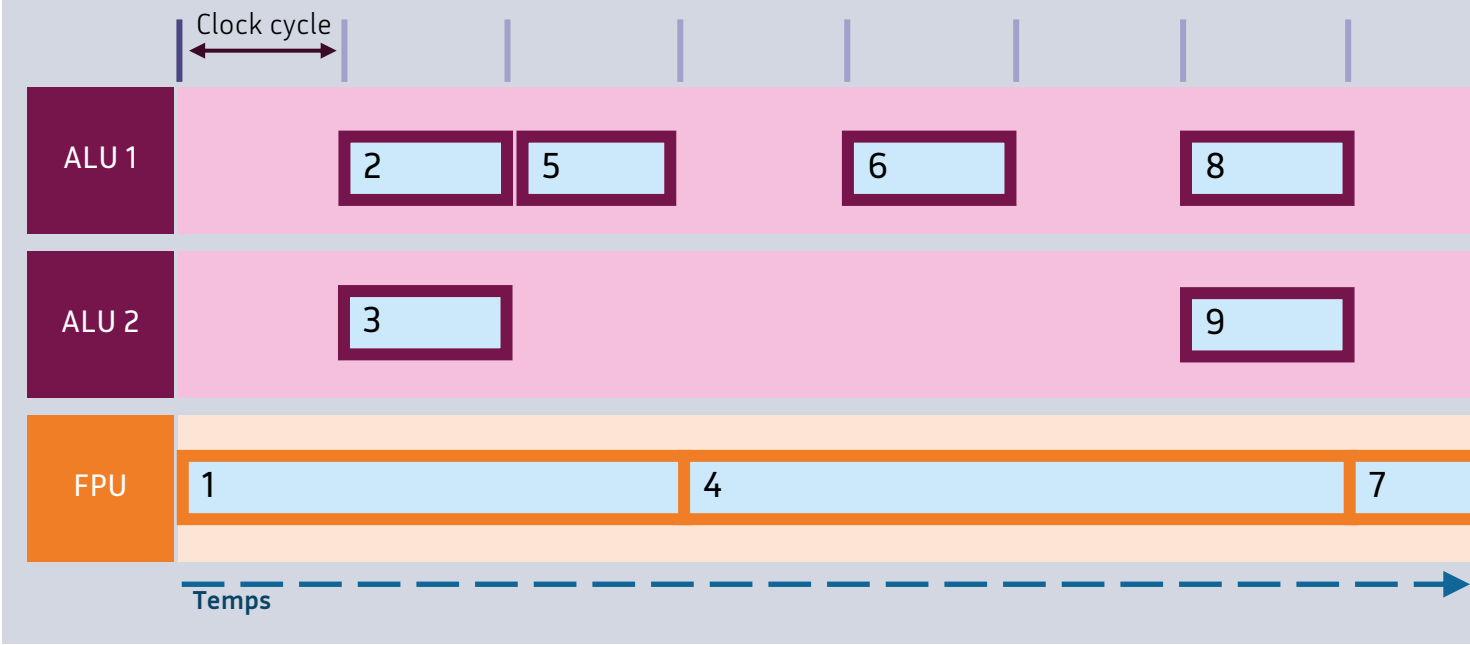


Superscalar Processors

Overlap instructions processing

- **Homogeneous instructions**
  - Multiple identical arithmetic units
  - Instruction re-ordering

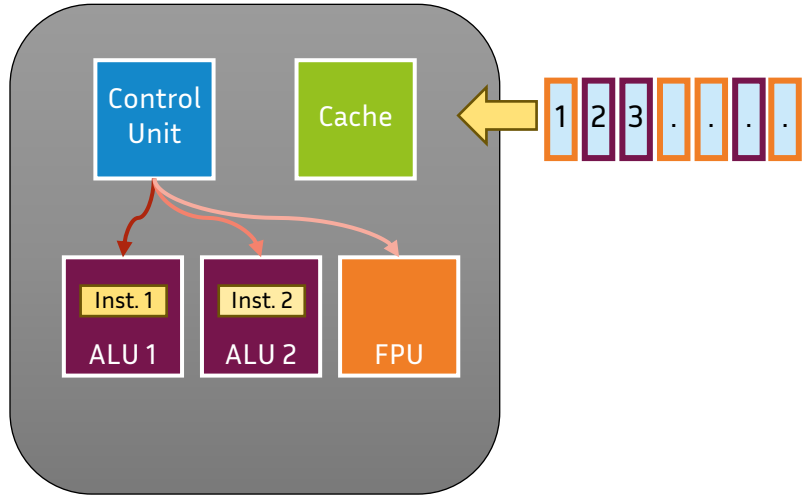


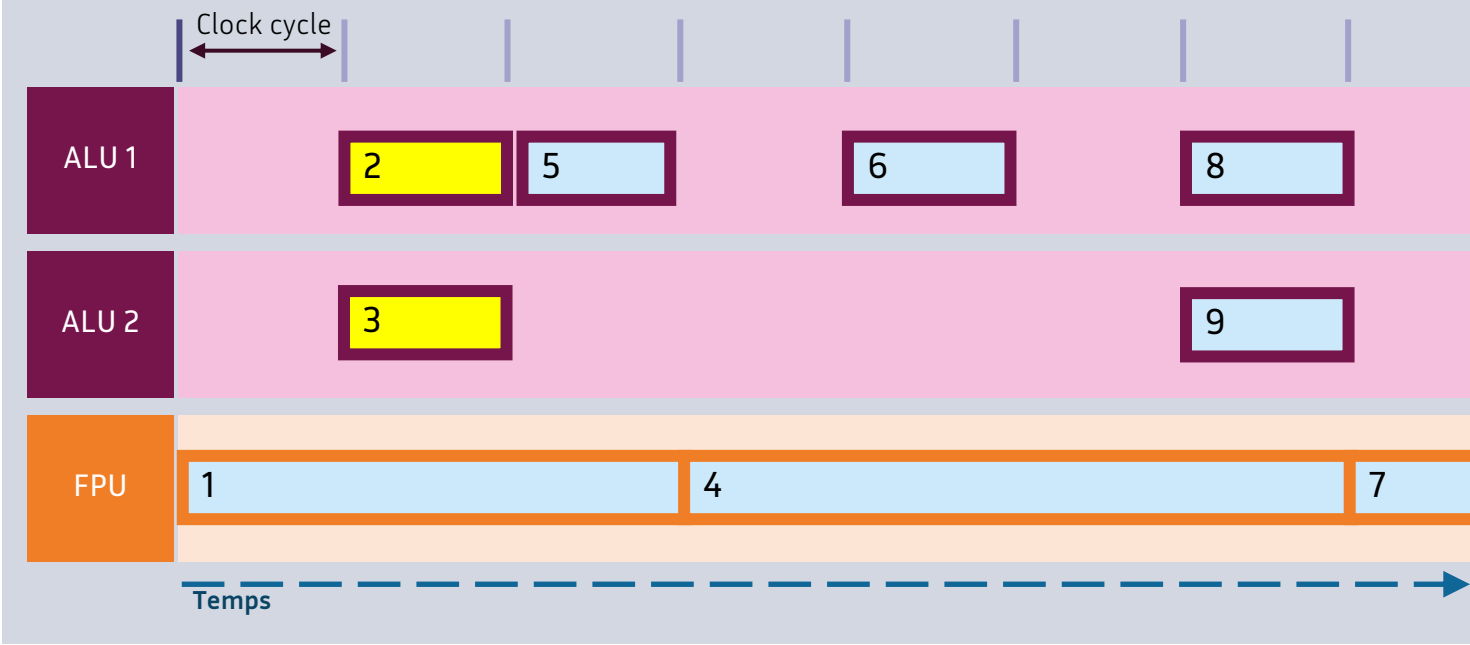


Superscalar Processors

Overlap instructions processing

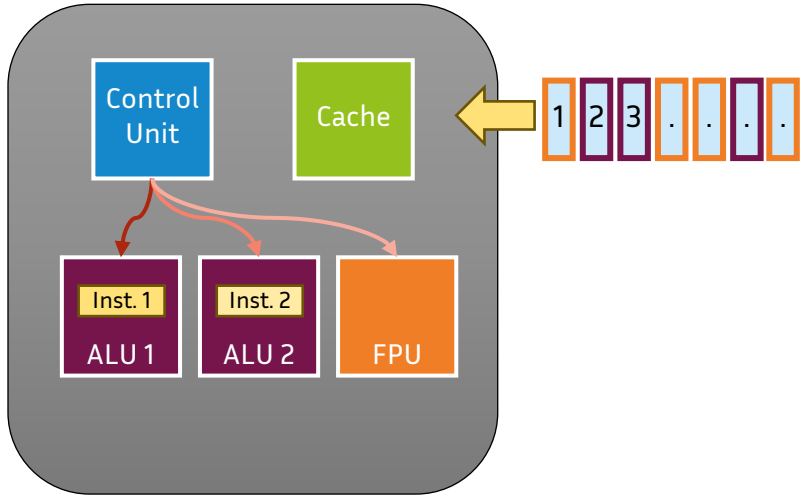
- **Homogeneous instructions**
  - Multiple identical arithmetic units
  - Instruction re-ordering



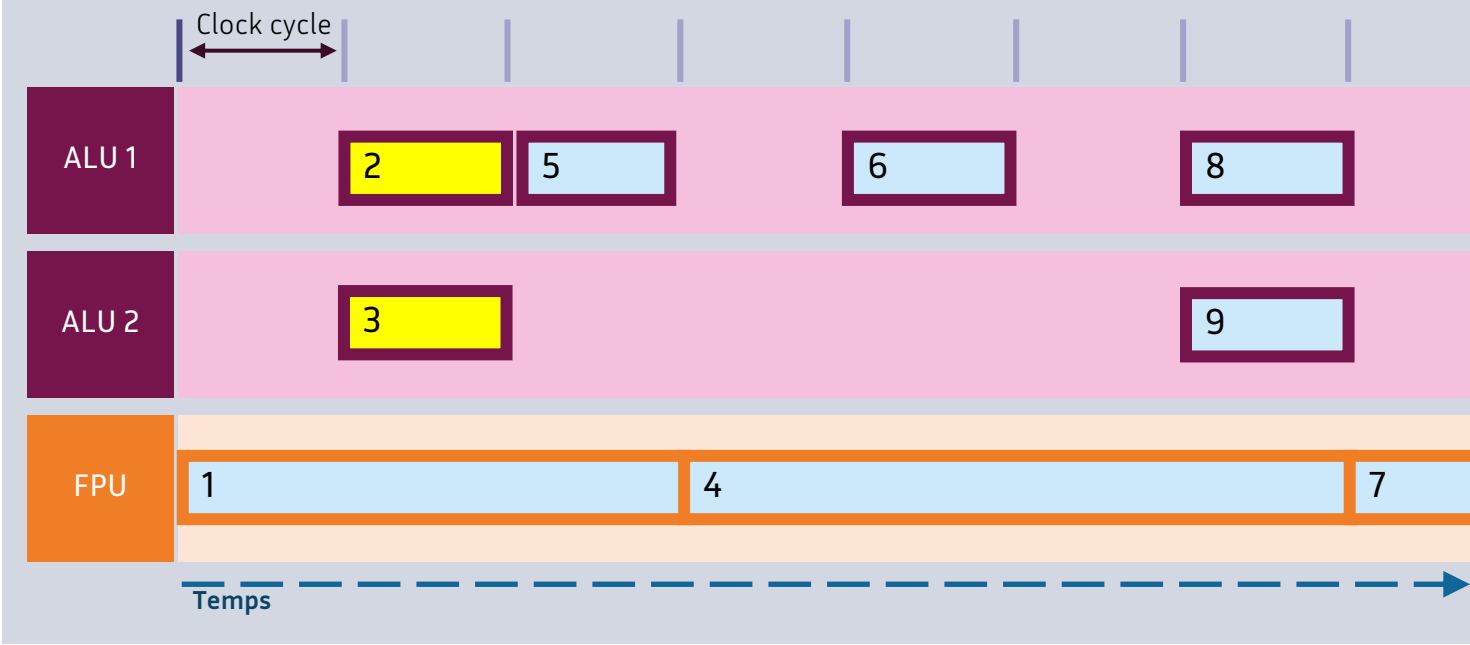


Superscalar Processors

Overlap instructions processing



- **Homogeneous instructions**
  - Multiple identical arithmetic units
  - Instruction re-ordering
- **Consistency**
  - Data dependencies



Instruction-Level Parallelism

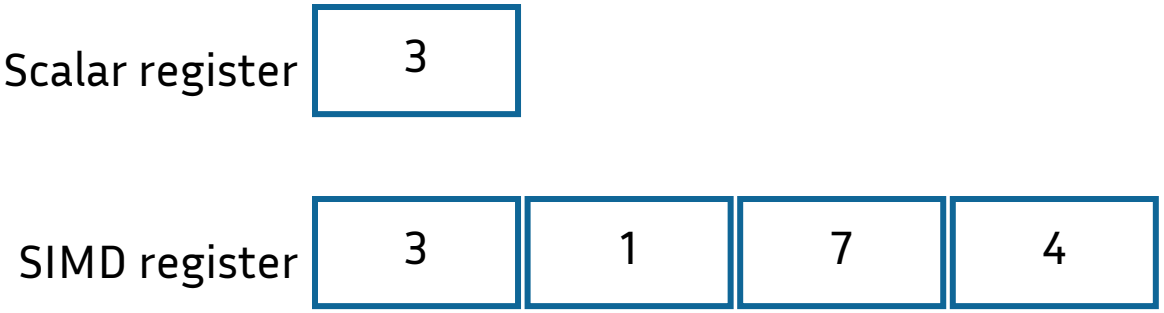
Extract parallelism within a single flow of instructions

- **Overlap the processing of successive instructions**
  - Pipelining
- **Process several instructions at a given time**
  - Overlapping / offload
  - Superscalar processing
- **Apply an instruction on multiple data**
  - SIMD instruction sets

SIMD Instruction Sets

Single Instruction Multiple Data

- **General principle**
  - SIMD register == small vector

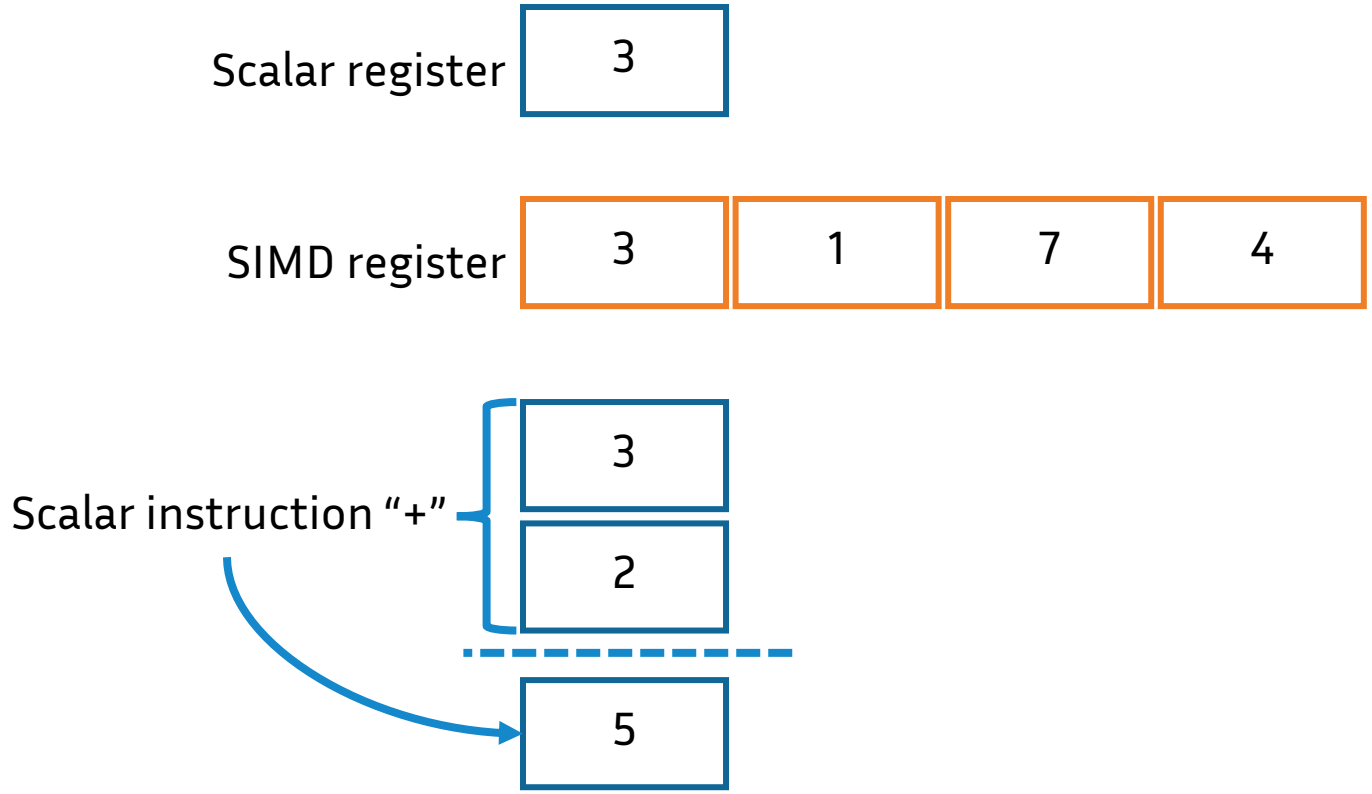


Typical RISC instruction processing sequence

SIMD Instruction Sets

Single Instruction Multiple Data

- **General principle**
  - SIMD register == small vector
  - SIMD arithmetic instructions
    - Ops applied to vector items…
    - … in parallel

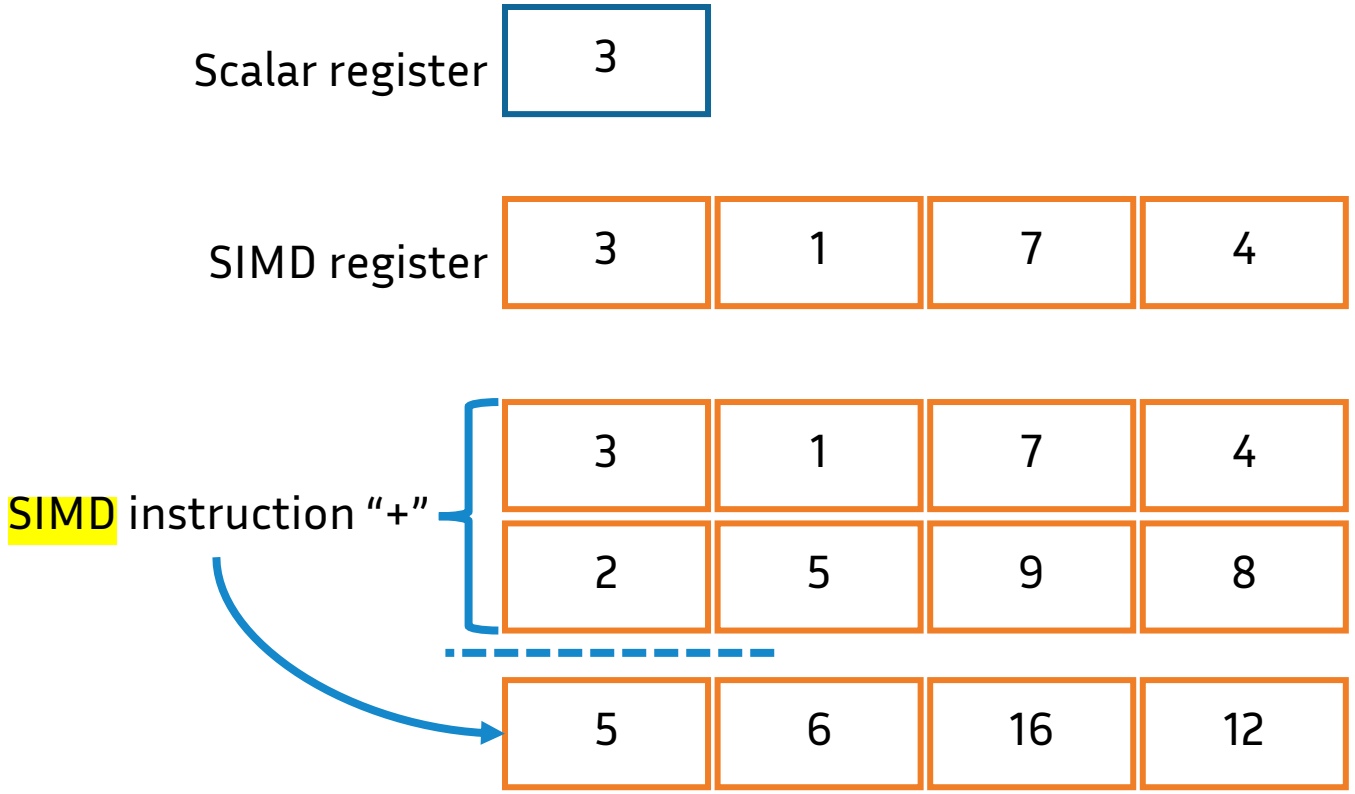


Typical RISC instruction processing sequence

SIMD Instruction Sets

Single Instruction Multiple Data

- **General principle**
  - SIMD register == small vector
  - SIMD arithmetic instructions
    - Ops applied to vector items…
    - … in parallel



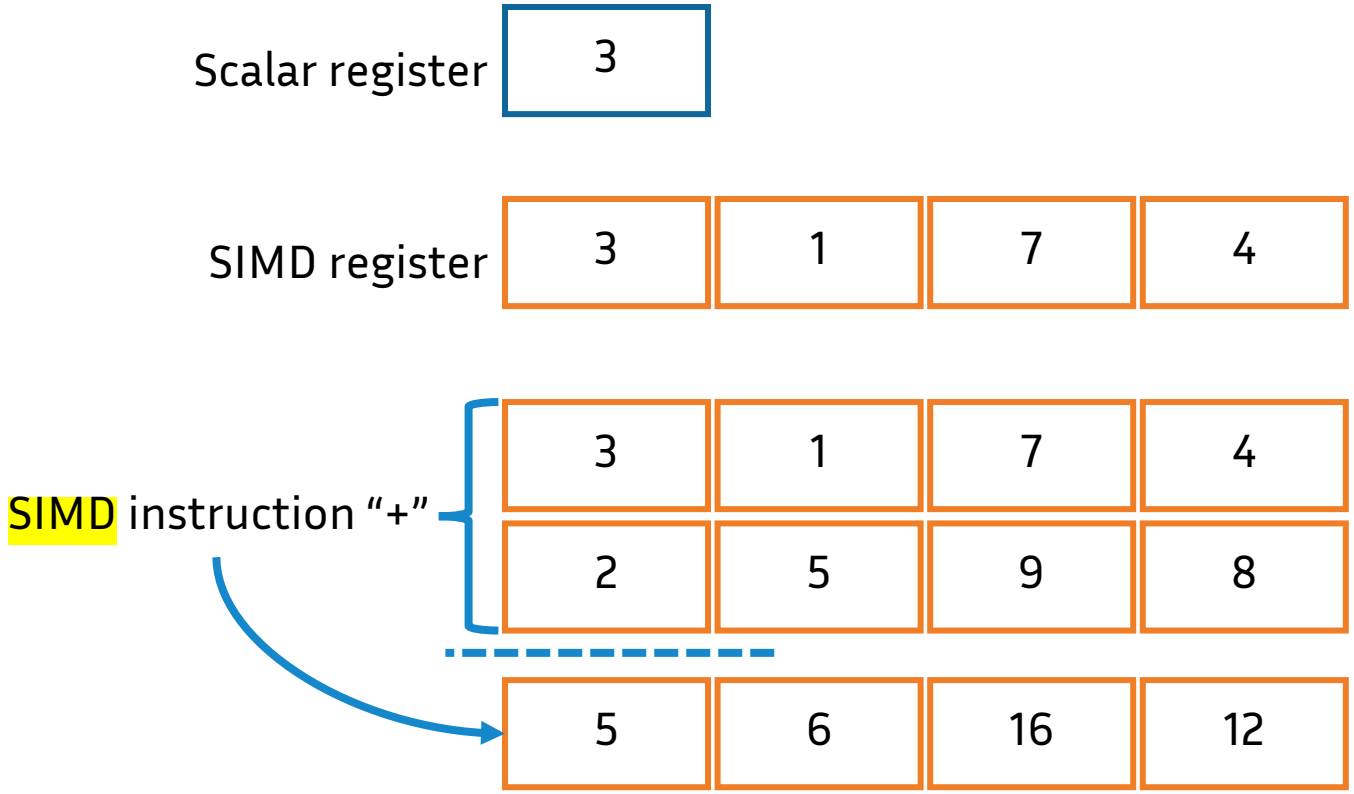
Typical RISC instruction processing sequence

SIMD Instruction Sets

Single Instruction Multiple Data

- **General principle**
  - SIMD register == small vector
  - SIMD arithmetic instructions
    - Ops applied to vector items…
    - … in parallel

- **Example**
  - Intel Intrinsics Guide [link]

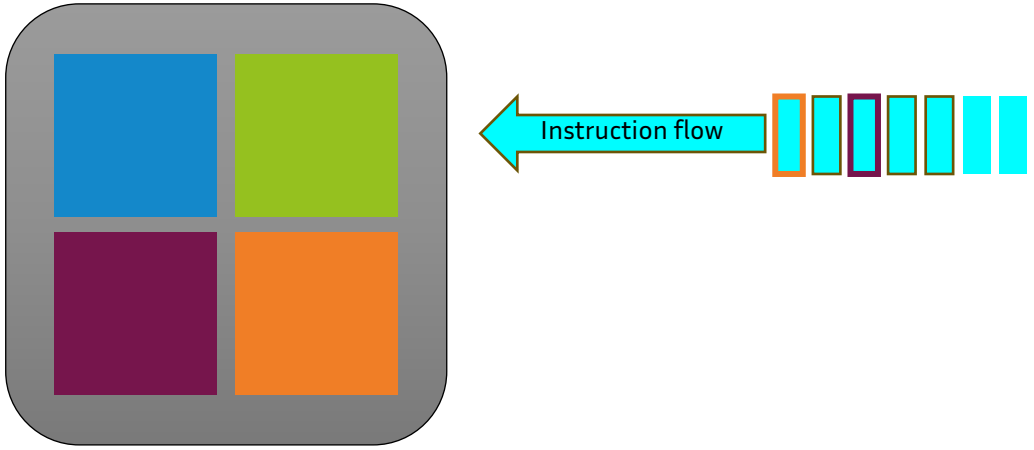


Typical RISC instruction processing sequence

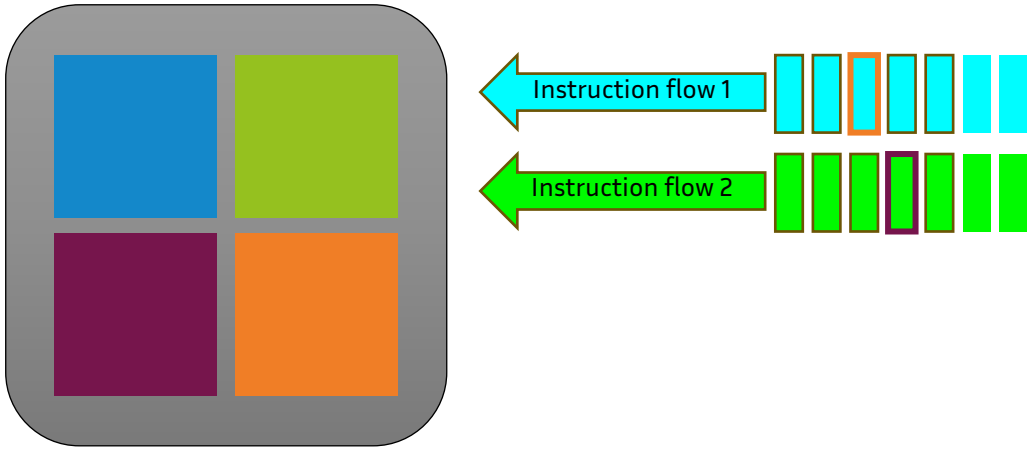
Doing More by Unit of Time

Hardware parallelism

- **Instruction-level parallelism (ILP)**
  - Parallelism within a single instruction flow



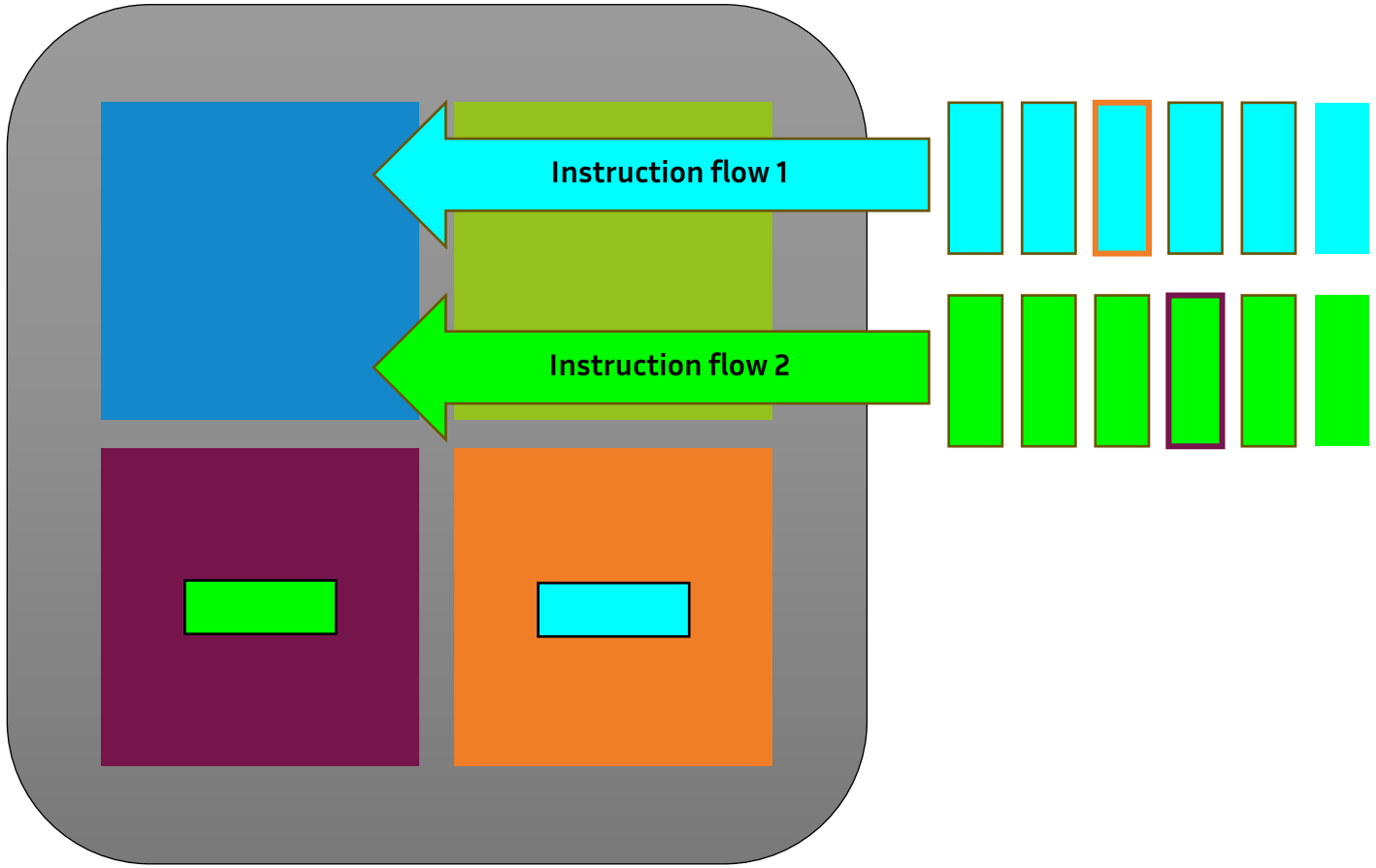
- **Thread-level parallelism (TLP)**
  - Parallelism across multiple instruction flows



Thread-Level Parallelism

Extract parallelism from multiple flows of instructions

- **No automatism**
  - Programs must expose multiple flows

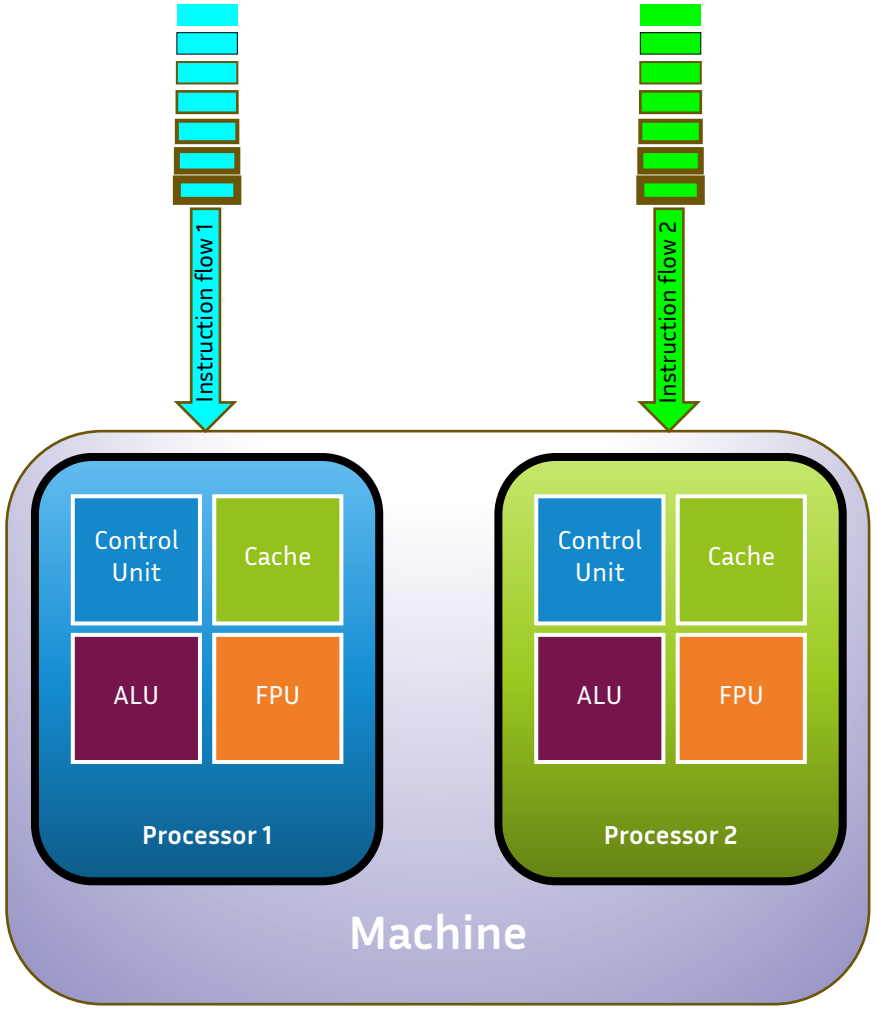


Parallel Architectures

Multiprocessor Architectures

Several processors in the same machine

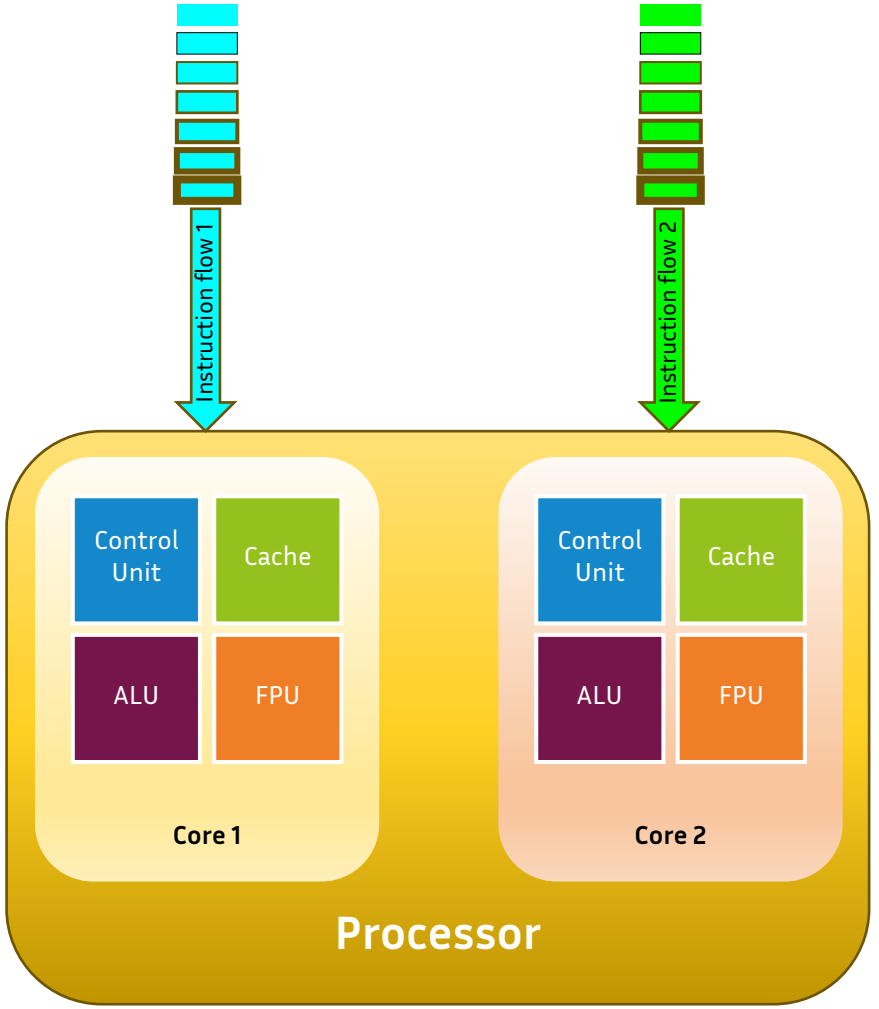
- **Considerations**
  - Benefits
  - Cost
  - Scalability
  - Memory



Multicore Architectures

Several processing areas in the same processor

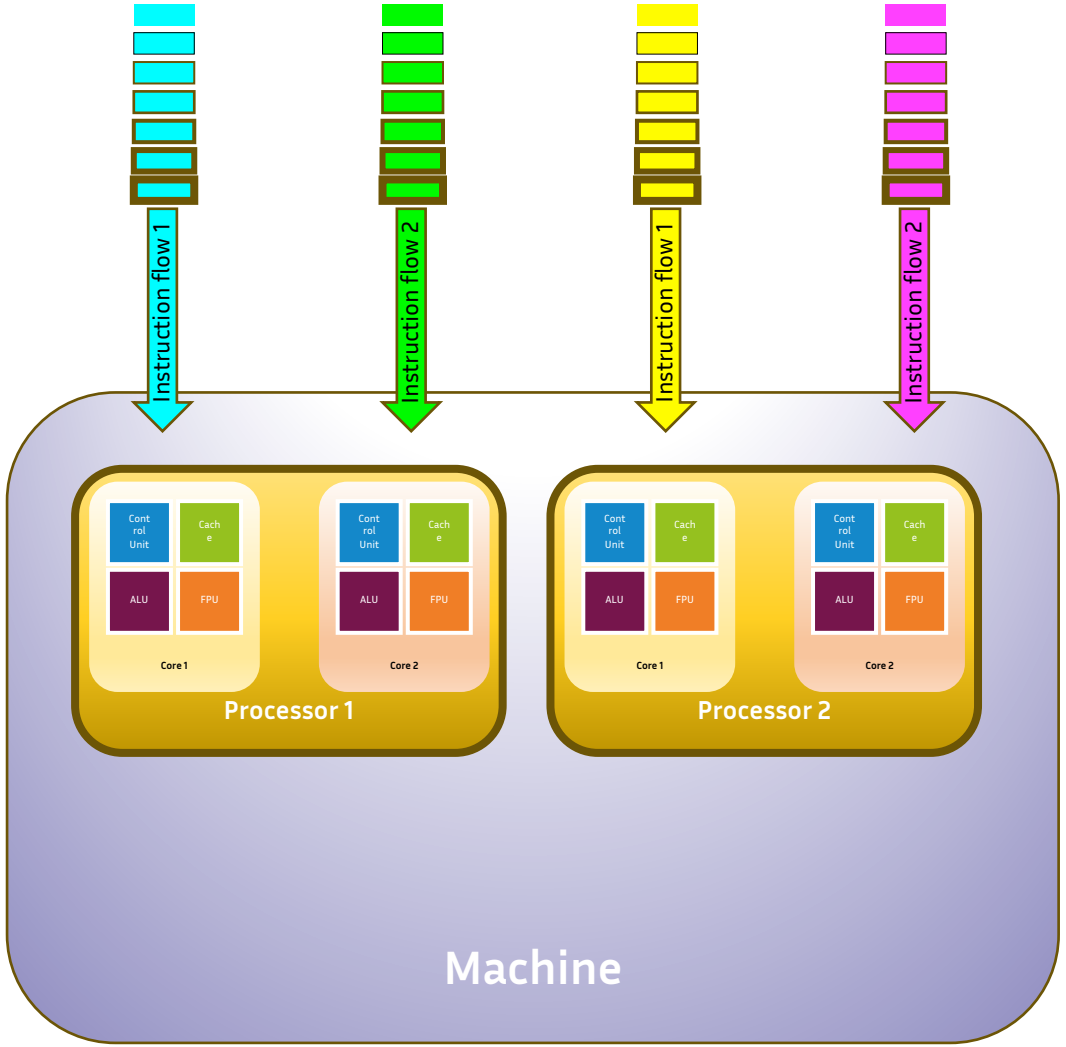
- **Considerations**
  - Benefits
  - Cost
  - Scalability
  - Memory



Multicore + Multiprocessor Architectures

Combining the two approaches

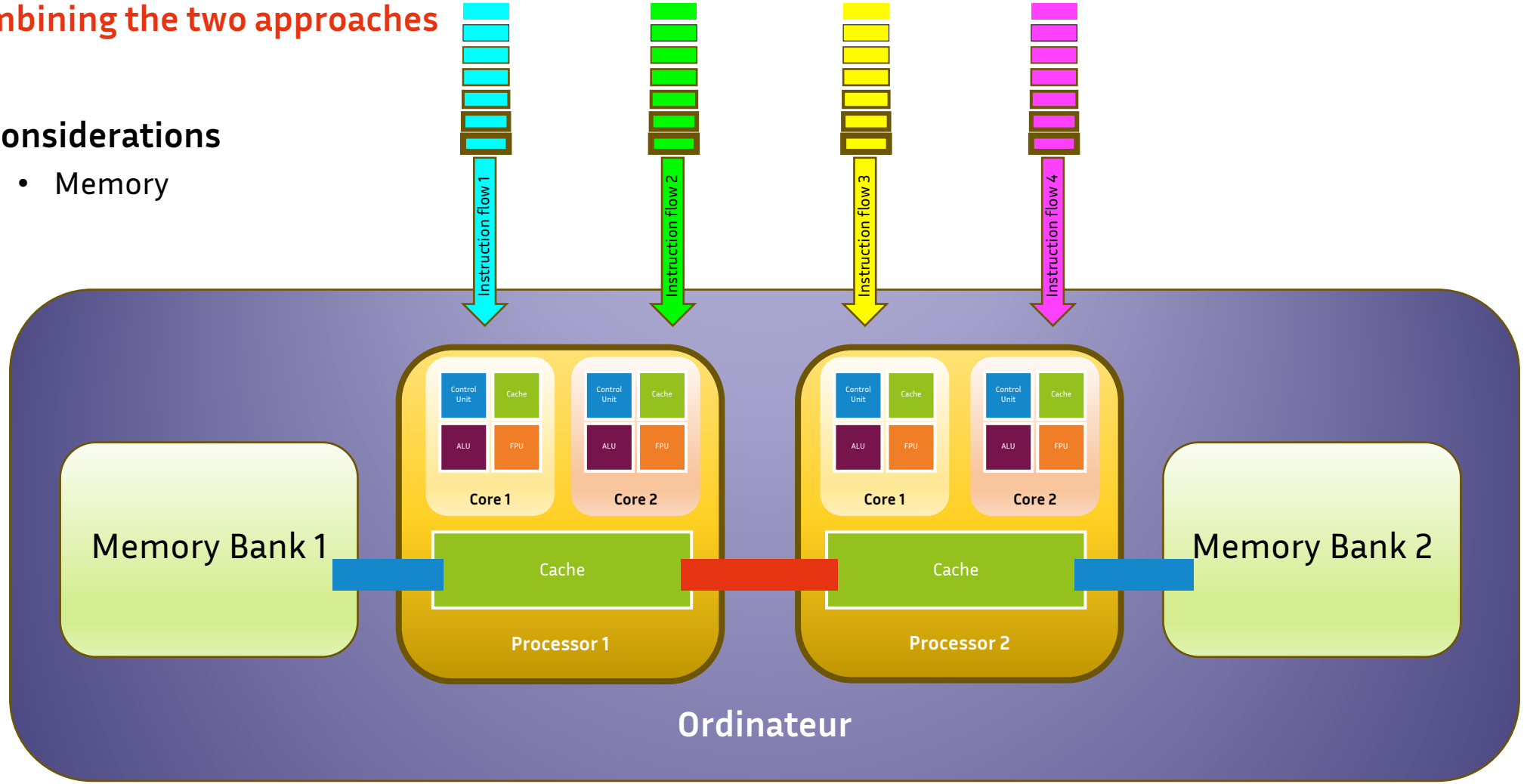
- **Considerations**
  - Benefits
  - Cost
  - Scalability



Multicore + Multiprocessor Architectures

Combining the two approaches

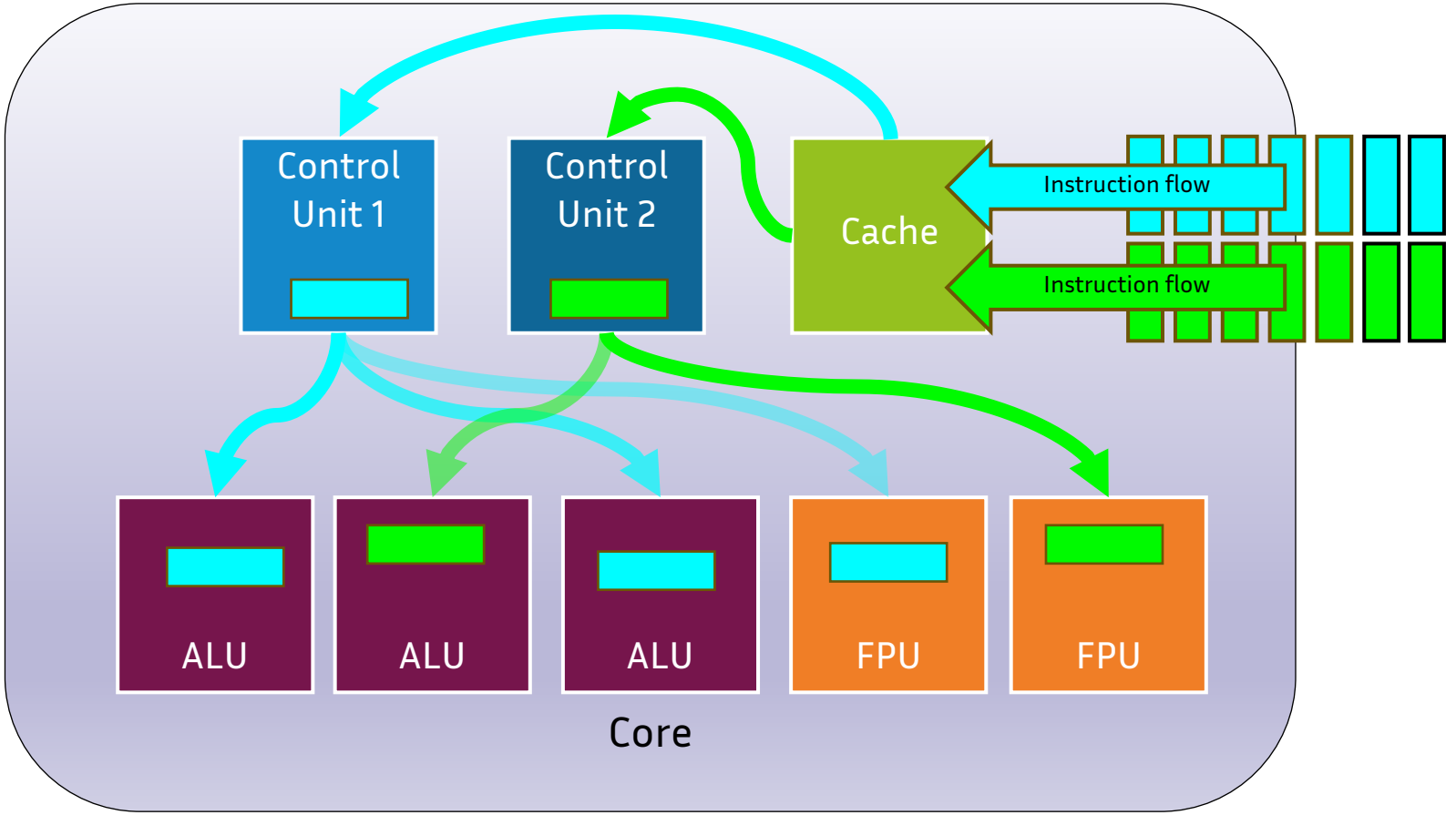
- **Considerations**
  - Memory



Hardware Multithreading

Feeding idle units

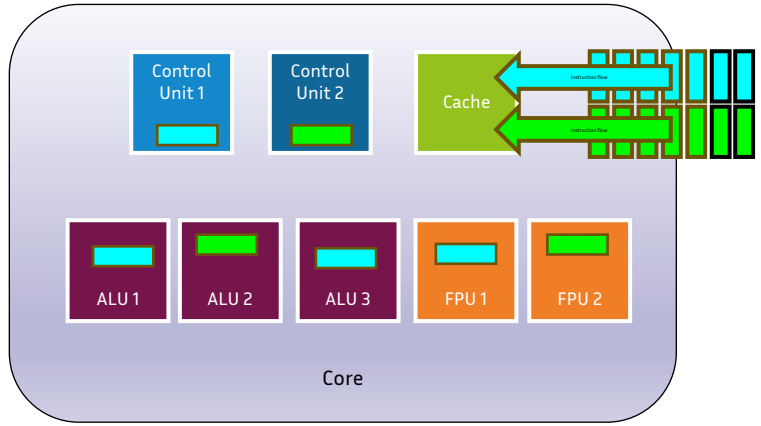
- **Principles**
  - Multiple thread contexts…
  - … managed in a single core

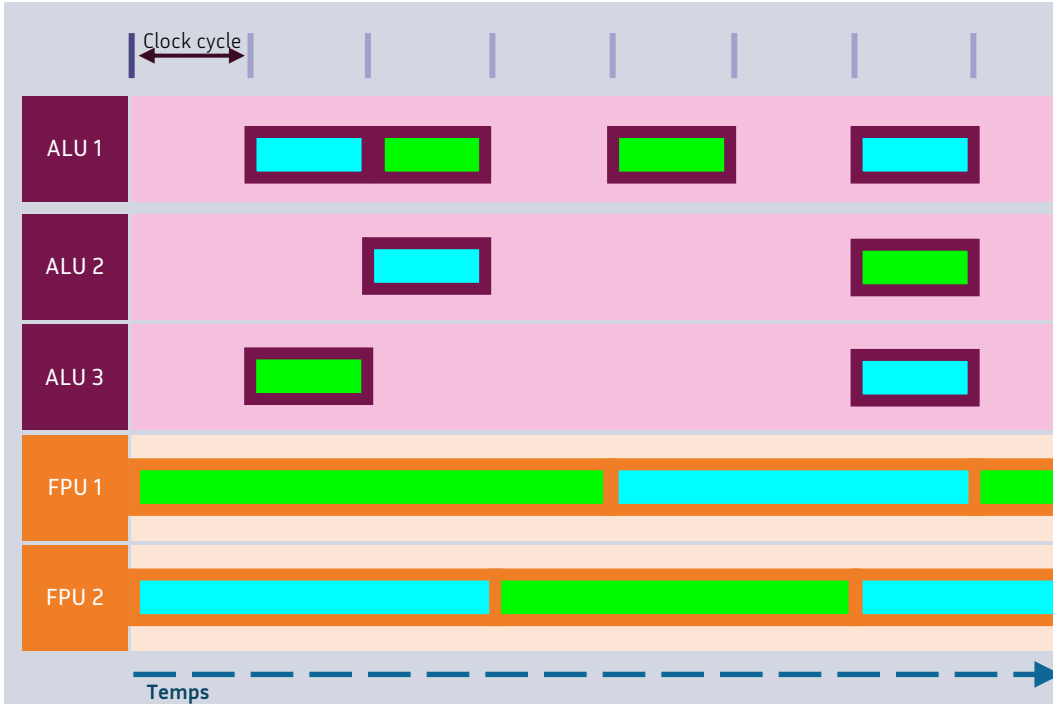


Hardware Multithreading

Feeding idle units

- **Principles**
  - Multiple thread contexts…
  - … managed in a single core
- **Considerations**
  - Benefits
  - Drawbacks

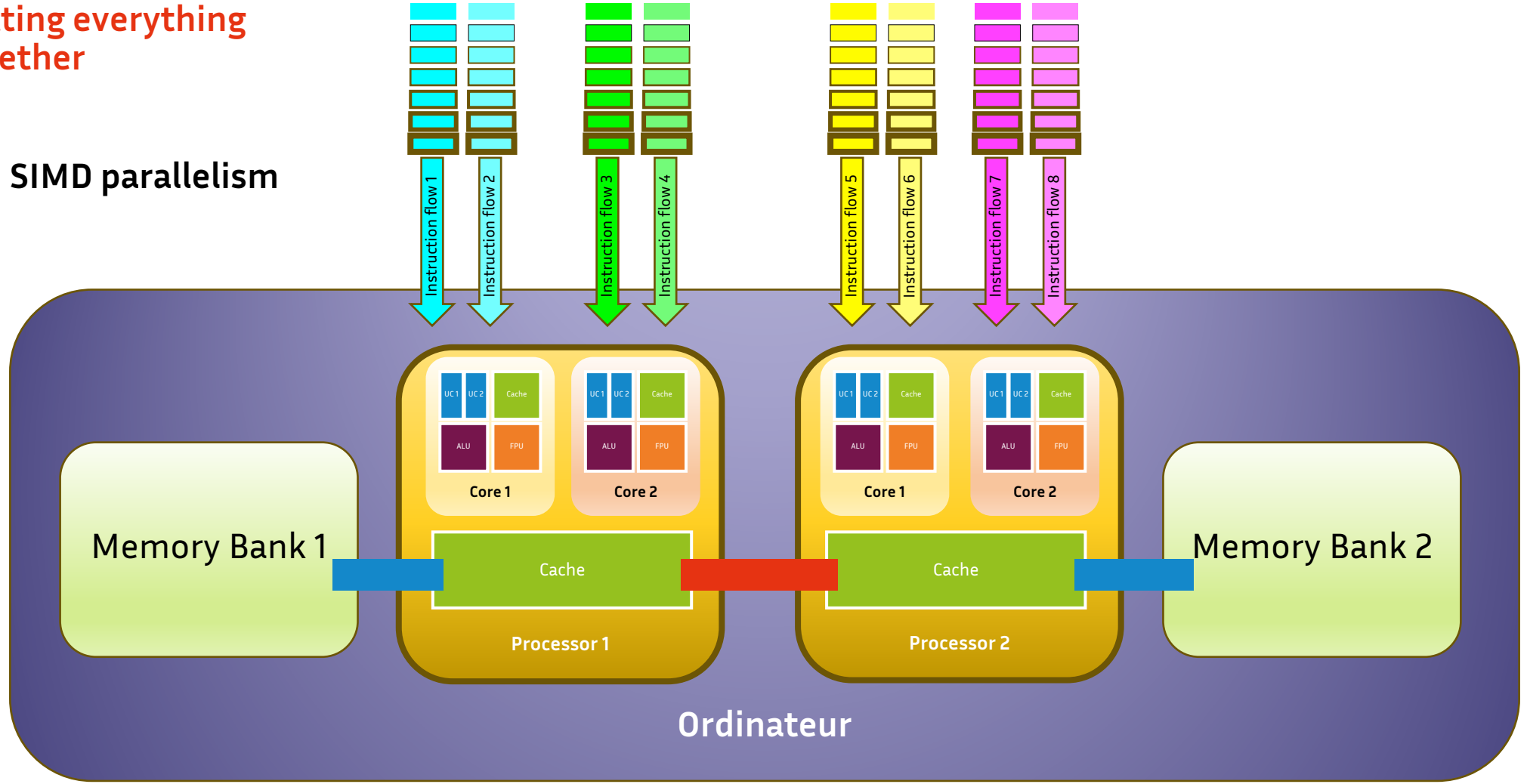




HW MT + Multicore + Multiprocessor Architectures

Putting everything together

- **+ SIMD parallelism**



TD

Topology discovery with libhwloc

- **Assignment, source codes, additional slides**
  - Moodle ENSEIRB, IT390 course
- **Libhwloc on OpenMPI website**
  - [link]
- **Command-line tools**
  - lstopo
  - hwloc-bind
  - hwloc-calc

- **Keywords**
  - Topology
  - Affinity
  - Binding