

# INTENSIVE COMPUTATION

---

**Annalisa Massini**  
2022-2023

*Lecture 1*

# COURSE INFORMATION

---

# Course topics

Since last year, the course has been modified: some concepts of quantum computing have been included, while some topics have been reduced or dropped

**This year's course** will focus on:

- Advanced topics of **computer architectures**
- **Arithmetic circuits** and their evaluation
- **Interconnection topologies** for HPC and PD systems and related communication problems
- Introduction to **Quantum Computing**, with particular regard to quantum arithmetic circuits

# Time, venue and course page

- **T1 room building E: Monday 14:00-17:00 - Thursday 11:00-13:00**
- Lectures will be given mainly by using slides or by using the blackboard/tablet
- Course page is:  
<http://twiki.di.uniroma1.it/twiki/view/CI/WebHome>
- Use the course page to get information on the lessons, access the course material and receive notices

# Exam

## Exam will consist of:

- Written part
  - Two partial exams (*Midterm + end-of-term*) or a final exam – *that will consist in a written test with exercises*
- Oral part
  - Oral exam **or** Project **or** Presentation of one-two papers
  - *NOTE that project topic and papers for presentation must cover one course topic and **must be approved by the teacher***
- It is possible that **homework** will be assigned during the course and will contribute to the final grade

# Books

I will not use a single reference book

I will give you:

- Reference books for each topic of the course
- Lectures slides

The main books I will refer to are:

- **Computer Architecture - A Quantitative Approach**

*J. L. Hennessy, D.A. Patterson – Morgan Kaufmann*

*5<sup>th</sup> Ed. 2011 and 6<sup>th</sup> Ed. 2019*

- **Dancing with qubits**

*R.S. Sutor – Packt> - 2019*

# Books

Some chapters from the following books:

***Advanced Computer Architecture and Parallel Processing***

*H. El-Rewini, M. Abd-El-Barr, John Wiley and Sons, 2005*

***Parallel computing for real-time signal processing and control***

*M. O. Tokhi, M. A. Hossain, M. H. Shaheed – Springer – 2003*

***Multicore and GPU Programming An Integrated Approach***

*G. Barlas – Morgan Kaufmann – 2014*

***Parallel Computer Architecture: A Hardware/Software Approach***

*D.E. Culler, J. P. Singh, A. Gupta – Morgan Kaufmann – 1998*

***Introduction to High Performance Computing for Scientists and Engineers***

*G. Hager G. Wellein – CRC Press – 2011*

***Programming Massively Parallel Processors***

*D.B. Kirk W. W. Hwu - Morgan Kaufmann – 2013*

# Course topics

## Part 1 – Architectures

- Overview on the Von Neumann architecture
- Circuits for arithmetic operations and circuit evaluation
- Number representations for Fast Arithmetic
- Motivation to parallel architectures and their classifications
- SIMD class: vector architecture and GPUs
- Sparse matrices: compact storage methods



# Course topics

## Part 2 – Parallel architectures and quantum circuits

- Performance metrics and measurements for computer architectures evaluation
- MIMD class: interconnection networks and related problems
- Quantum Computing and quantum arithmetic circuits

## Some possible topic (if we will have time)

- Errors
- Methods for solving linear systems
- Eigenvalues and eigenvectors: methods and applications.
- Molecular Dynamics: Hooke's Law Model, Lennard-Jones Model, Hard Sphere Model

# INTRODUCTION

---

*The importance of high-performance architectures  
and their impact on computational science*

# Introduction

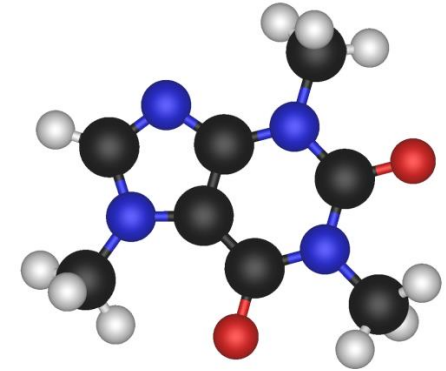
- Traditional **methods in science** and **engineering** are:
  - To develop *theories* and *projects*
  - To execute *experiments* and
  - To build *systems*
- The realization of these tasks can be :
  - Too **difficult** → wind tunnel
  - Too **expensive** → crash testing
  - Too **slow** → the evolution of a galaxy
  - Too **dangerous** → drugs, toxic gas diffusion



Chlorine release, 2010 Jack Rabbit I Program

# Introduction

- **Computers** represent the fundamental tool for the simulation and can be seen both as a **microscope** and as a **telescope** with respect to space and to the time
- **Examples**
  - Model molecules in details
  - Travel to the origin of the universe and study its evolution
  - Provide weather forecasts or climate changes



# Introduction

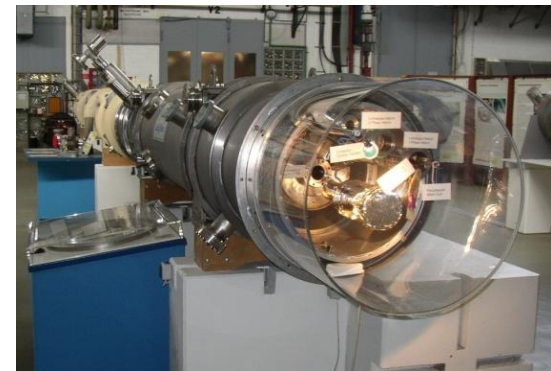
Instruments as particle accelerator, telescopes, scanner, etc., produce big quantity of data

**Data** are elaborated by a computer and are:

- Reduced and transformed
- Represented and visualized

Objectives of **data elaboration** are:

- To understand the meaning of the produced data
- To develop new theories
- To verify different kind of phenomena



# Computational Science

- Computational science **is concerned with**:
  - Mathematical models
  - Quantitative analysis techniques
  - **Computer** elaboration
  - Analysis and solution of scientific problems
- Computational science **involves**:
  - The application of **computer** simulation
  - Different forms of computation (numerical analysis, theoretical computer science, etc.)
  - Problems in various scientific disciplines

$$\frac{dLs}{dt} = N * K_1 - (d_1 + r) * Ls + \delta_1$$

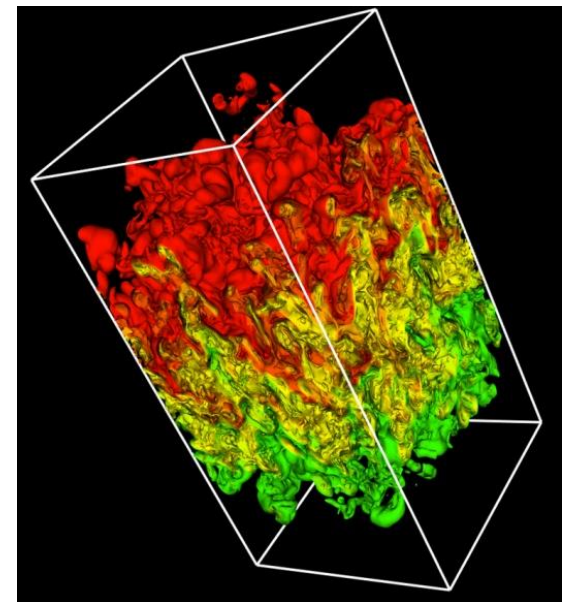
$$\frac{dLux}{dt} = \left( \frac{K_2}{1 + A * Ls^2} \right) * N - (d_2 + r) * Lux + \delta_2$$

$$Z = Z_1 + Z_2$$

$$Z_1 = K_3 * Lux$$

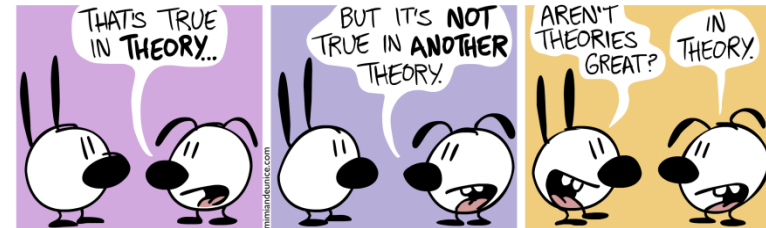
$$Z_2 = \iint_{0,0}^{2\pi,D} e^{-K_4 s} * Z_{i,j} ds$$

$$\frac{dLs1}{dt} = Ls * \left( 1 - \frac{Ls}{K5} \right) * Z * \left( 1 - \frac{Z}{K6} \right) \quad (1)$$



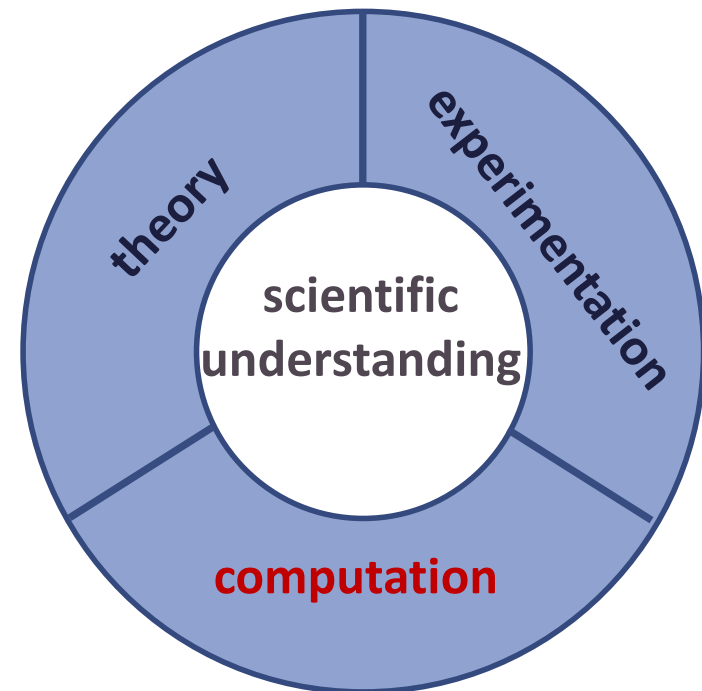
# Computational Science

- The **scientific computing approach** is to gain understanding, mainly through the analysis of mathematical models implemented on **computers**



## Computational science:

- Is different from **theory** and **laboratory experiments**, traditional forms of science and engineering
- Is now considered a **third mode of science**, besides theory and experimentation/observation





# Computational Science

- Scientists and engineers develop **computer** programs and application software, that model systems being studied
- These programs are run with various sets of input parameters
- In most cases, these models require **massive amounts of calculations** (usually floating-point numbers) that are executed on **supercomputers** or **distributed computing systems**





# FUNDAMENTALS OF QUANTITATIVE DESIGN AND ANALYSIS

---

***Computer Architecture: A Quantitative Approach***

J. L. Hennessy, D. A. Patterson - Morgan Kaufmann, 2012

# Computer architectures evolution

- The dramatic **growth rate in computer performance** in the 20th century has been **fourfold**
- **First impact**
  - Significant enhancement of the **capability available to computer users**
  - For many applications, the highest-performance microprocessors of today outperform the supercomputer of less than 20 years ago

# Computer architectures evolution

- **Second impact**
  - New classes of computers
  - In the 1980s: *personal computers* and *workstations* thanks to microprocessor
  - Last two decades: *smart cell phones* and *tablet computers*, used as primary computing platforms instead of PCs, exploiting the *Internet* to access warehouses containing *tens of thousands of servers*, as they were a single gigantic computer

# Computer architectures evolution

- **Third impact**

- Improvement of semiconductor manufacturing (predicted by Moore's law) led to the **dominance of microprocessor-based computers**
- Minicomputers, traditionally made from gate arrays, were replaced by **servers** made using **microprocessors**
- Even **mainframe computers** and **high-performance supercomputers** are all collections of **microprocessors**
- The hardware innovations led to a **renaissance in computer design**, which emphasized both **architectural innovation** and **efficient use of technology improvements**
- By 2003, high-performance microprocessors were 7.5 times faster than what would have been obtained by relying solely on technology, including improved circuit design
- That is, 52% per year versus 35% per year

# Computer architectures evolution

- **Fourth impact**

- The hardware renaissance had impact on **software development**
- In place of performance-oriented languages like C and C++, much more programming today is done in managed programming languages like **Java** and **C#**
- Scripting languages like **Python** and **Ruby**, which are even more productive, gained in popularity
- To maintain productivity, **interpreters with just-in-time compilers** and trace-based compiling are replacing the traditional compiler + linker
- Software deployment is changing as well: **Software as a Service** (SaaS) used over the Internet instead of software running locally
- The **nature of applications** is also changing: speech, sound, images, and video are becoming increasingly important, along with predictable response time that is so critical to the user experience

# Computer architectures evolution

- At a certain point, the **hardware renaissance seemed to be over**
- In 2004 Intel canceled its high-performance uniprocessor projects and joined others in declaring that the road to higher performance would be via ***multiple processors per chip rather than via faster uniprocessors***
- There has been a historic shift from relying solely on the **instruction-level** parallelism (ILP) to **data-level** parallelism (DLP) and **thread-level** parallelism (TLP)
- Now also the **warehouse-scale computers** and the **request-level** parallelism (RLP) need to be considered

# GENERATION OF COMPUTERS AND CLASSES

---

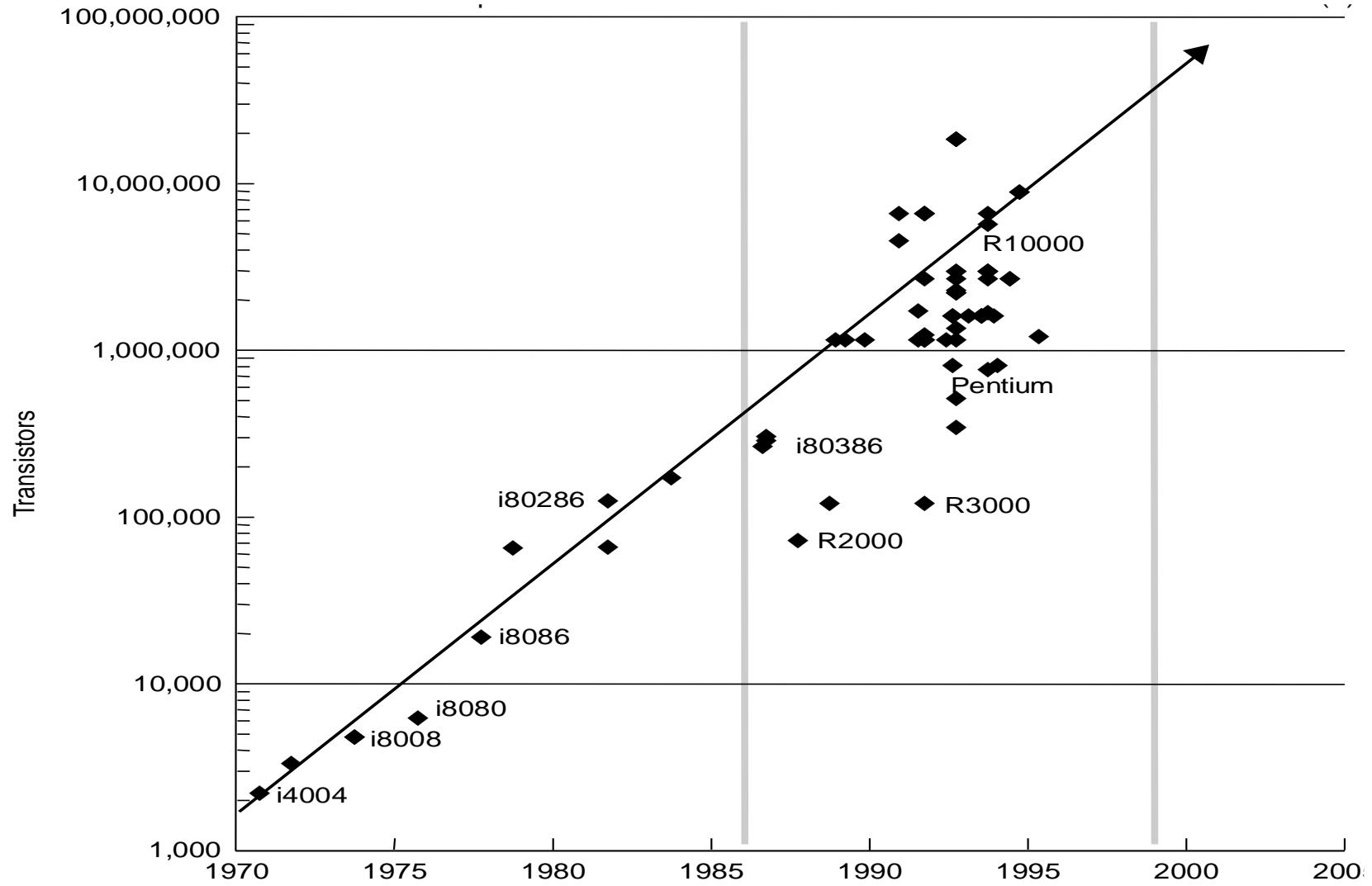
# Generations of Computers

- The history of computer architecture is traditionally divided into **four generations** (basic logic technology):
  - **1 - Vacuum tube** - 1946-1957
  - **2 - Transistor** - 1958-1964
  - **3 - Integrated circuits**
    - Small scale integration* - 1965 on  
Up to 100 devices on a chip
    - Medium scale integration* - to 1971  
100-3,000 devices on a chip
    - Large scale integration* - 1971-1977  
3,000 -  $10^5$  devices on a chip
  - **4 - VLSI**
    - Very large scale integration* - 1978-1991  
 $10^5$  -  $10^8$  devices on a chip
    - Ultra large scale integration* - 1991-  
Over  $10^8$  devices on a chip



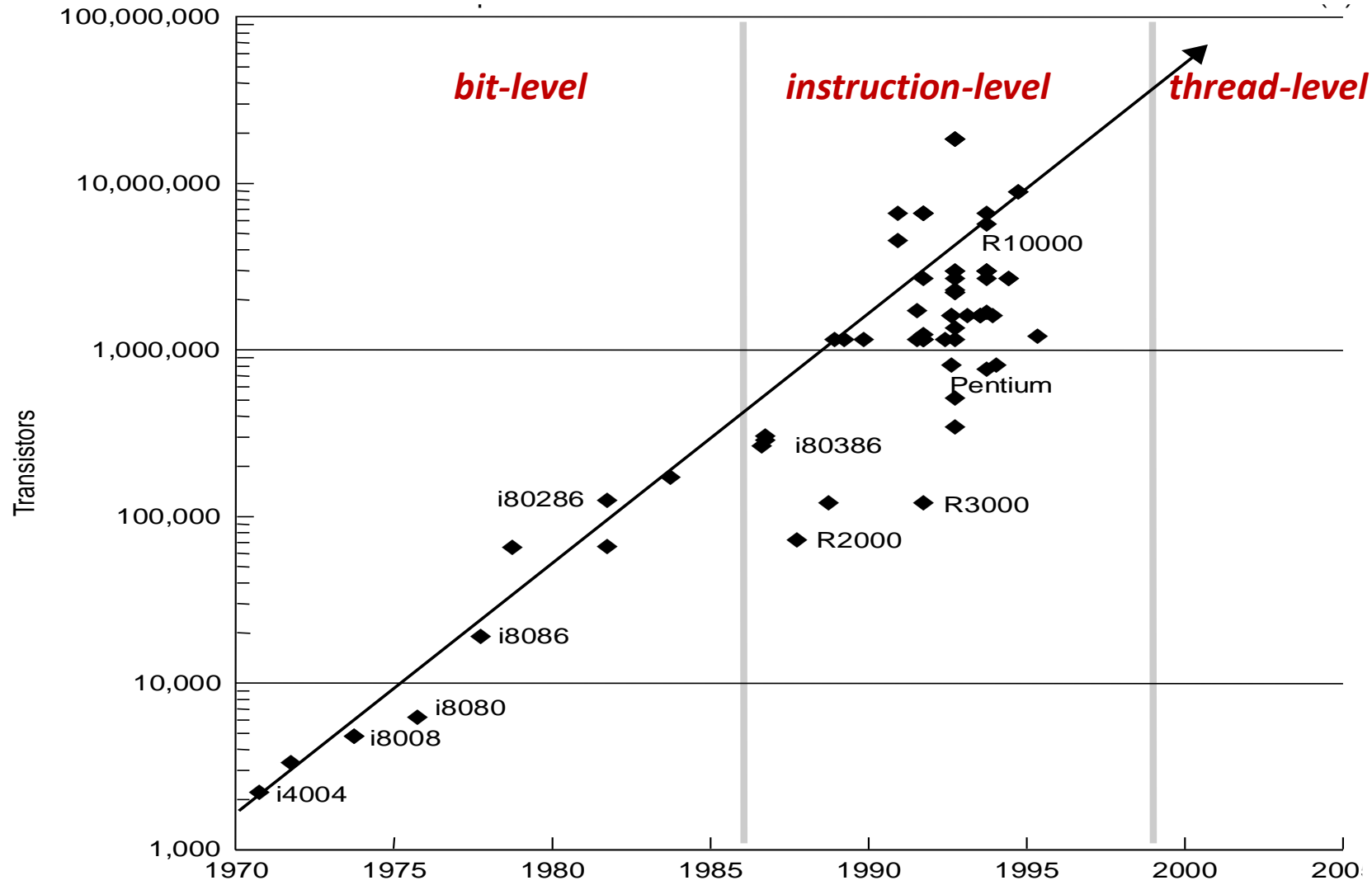
# Architectural Trends

**4th generation** - VLSI generation - presents a great architectural advance



# Architectural Trends

The strongest delineation in VLSI generation is the *kind of parallelism* exploited



# Classes of Computers

- The changes described
  - have influenced the **computer markets** in this new century
  - And, at the same time, **our vision of computing and computer applications** has changed considerably
- Since the creation of the personal computer, there have been no **changes** as evident as they are today in the **appearance** of computers and in the **way they are used**
- These changes in computer use have led to **five different computing markets**, each characterized by different *applications, requirements, and computing technologies*

# Classes of Computers – 1

- **Personal mobile device** (PMD) is the term applied to a collection of wireless devices with multimedia user interfaces such as *cell phones, tablet computers*, and so on
- **Cost** is a prime concern determining the consumer price
- **Energy efficiency** is driven by both battery power and heat dissipation
- The **memory** can be a substantial portion of the system cost, and it is important to **optimize memory size**
- The importance of **memory size** translates to an emphasis on **code size**, since data size is dictated by the application

# Classes of Computers – 2

- **Desktop computing** spans from **low-end netbooks**, sold at low price, to high-end, heavily configured **workstations** that are more expensive
- Since 2008, more than half of the desktop computers made each year have been battery operated **laptop computers**
- The desktop market tends to be driven to optimize *price-performance*:
  - **performance** is measured primarily in terms of compute performance and graphics performance
  - **price** is what matters most to customers in this market, and hence to computer designers
- As a result, the newest, highest-performance and cost-reduced microprocessors often appear first in desktop systems

# Classes of Computers – 3

- Since the 1980s, the role of servers has grown
  - to provide larger-scale and more reliable file and computing services
  - replacing the traditional mainframe
- First key feature is **availability**:
  - most servers must operate seven days a week, 24 hours a day
  - a failure can be catastrophic – consider, e.g., the servers running ATM machines for banks or airline reservation systems
- A second key feature is **scalability**:
  - In fact, server systems grow in response to an increasing demand for the services they support or an increase in functional requirements
- Finally, servers are designed for **efficient throughput**, and the overall performance of the server is what is crucial

# Classes of Computers – 4

- The growth of the class of computers called **clusters** is due to the growth of Software as a Service (SaaS) for many applications
- Clusters are **collections of desktop computers or servers** connected by *local area networks* to act as a single larger computer
- The largest of the clusters are called **warehouse-scale computers** (WSCs), and tens of thousands of servers can act as one
  - Price-performance, power and availability are critical to WSCs
  - WSCs emphasize **interactive applications, large-scale storage, dependability, and high Internet bandwidth**
- **Supercomputers** or **HPC systems** are related to WSCs in that they are equally expensive
  - Supercomputers emphasize **floating-point performance and run communication-intensive batch programs** that can run for weeks

# Classes of Computers – 5

- **Embedded computers** are found in everyday machines: microwaves, washing machines, most printers, most networking switches, and all cars contain simple embedded microprocessors
- The **ability to run third-party software** is the dividing line between non-embedded and embedded computers, so as PMD is a different category with respect to embedded computers
- Although the range of computing power in the embedded computing market is very large, **price is a key factor** in the design of embedded computers
- Performance requirements do exist, of course, but the primary goal is often meeting the **performance need at a minimum price**, rather than achieving higher performance at a higher price



# Classes of Parallelism and Parallel Architectures

- **Parallelism at multiple levels** is now the driving force of computer design across all classes of computers, with **energy** and **cost** being the primary constraints
- There are basically two kinds of parallelism in applications:
  - **Data-Level Parallelism** (DLP) arises because there are many **data items** that can be **operated** on at the **same time**
  - **Task-Level Parallelism** (TLP) arises because **tasks of work** are created that can **operate independently** and largely in **parallel**

# Classes of Parallelism and Parallel Architectures

**Computer hardware** in turn can exploit these two kinds of application **parallelism** in **four major ways**:

1. **Instruction-Level Parallelism** exploits data-level parallelism at modest levels with compiler help using ideas like **pipelining** and at medium levels using ideas like **speculative execution**
2. **Vector Architectures and Graphic Processor Units (GPUs)** exploit data-level parallelism by applying a single instruction to a collection of data in parallel

# Classes of Parallelism and Parallel Architectures

Computer hardware in turn can exploit these two kinds of application parallelism in four major ways:

3. **Thread-Level Parallelism** exploits either data-level parallelism or task-level parallelism in a tightly coupled hardware model (shared memory systems) that allows for interaction among parallel threads
4. **Request-Level Parallelism** exploits parallelism among largely decoupled tasks specified by the programmer or the operating system

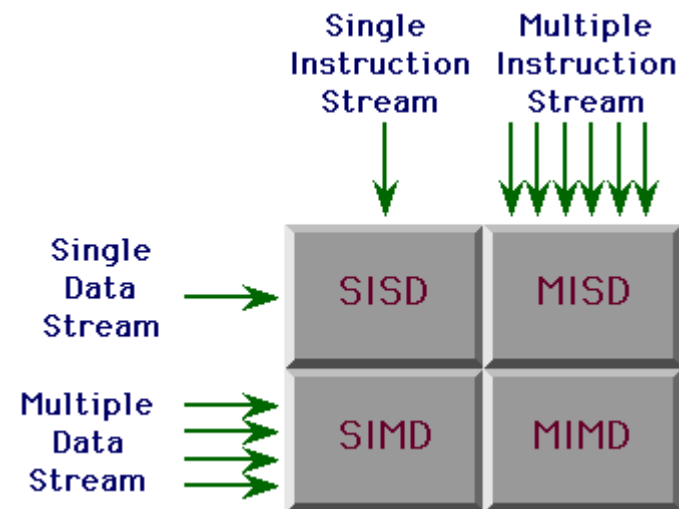
# Taxonomy of Computer Architectures

- These four ways for hardware to support the data-level parallelism and task-level parallelism go back to the 60s
- **Michael Flynn** studied the parallel computing efforts in the 60s, and introduced a **taxonomy** of computer architectures that is still the *most common way of categorizing* systems defining abbreviations we still use today
- He looked at the **parallelism** in the **instruction** and **data streams** called for by the instructions at the most constrained component of the multiprocessor, and placed all computers into one of four categories

# Taxonomy of Computer Architectures

- In Flynn's classification, machines are classified based on how many **data** items they can process concurrently and how many different **instructions** they can execute at the same time

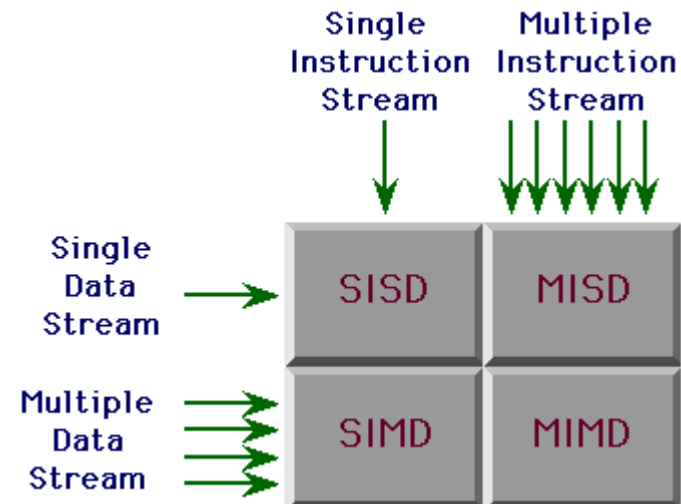
- Single Instruction, Single Data - **SISD**
- Single Instruction, Multiple Data - **SIMD**
- Multiple Instruction, Single Data - **MISD**
- Multiple Instruction, Multiple Data - **MIMD**



# Taxonomy of Computer Architectures

## Single Instruction stream, Single Data stream – SISD

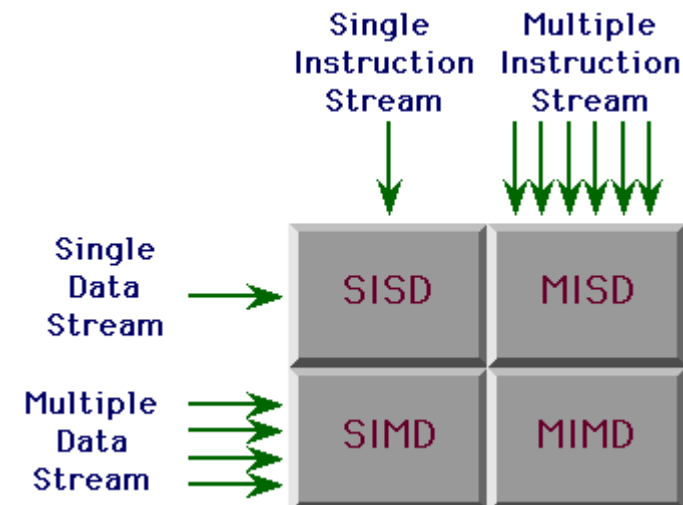
- This category is the **uniprocessor**
- The programmer thinks of it as the standard **sequential computer**, but it can exploit **instruction-level parallelism**
- SISD architectures use **ILP techniques** such as superscalar and speculative execution



# Taxonomy of Computer Architectures

## Single Instruction stream, Multiple Data stream - SIMD

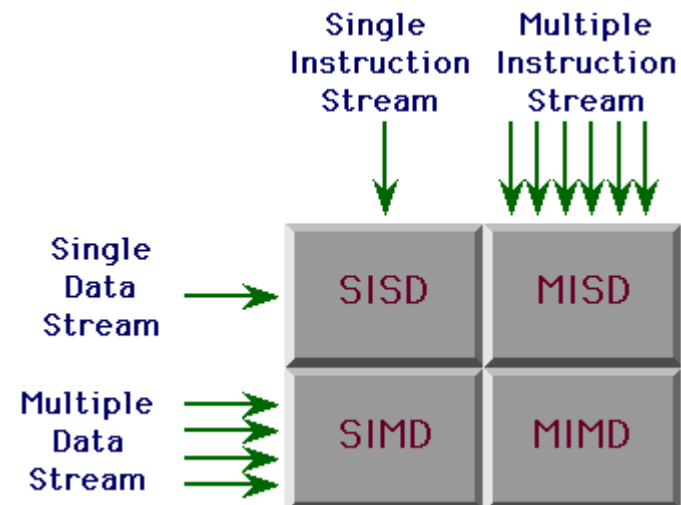
- The **same instruction** is executed by multiple processors using **different data streams**
- SIMD computers exploit **data-level parallelism** by applying the same operations to multiple items of data in parallel
- Each processor has its own data memory, but there is a single instruction memory and control processor, which fetches and dispatches instructions
- Examples are:
  - vector architectures
  - GPUs



# Taxonomy of Computer Architectures

## Multiple Instruction stream, Single Data stream – MISD

- **No commercial multiprocessor** of this type has been built to date
- But there can be possible future realizations/applications





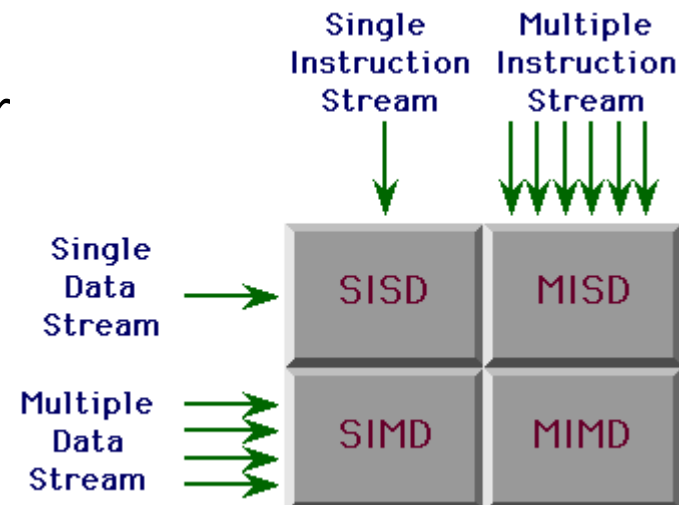
# Taxonomy of Computer Architectures

## Multiple Instruction stream, Multiple Data stream – MIMD

- Each processor fetches its own instructions and operates on its own data, and it targets **task-level parallelism**
- MIMD computers can also exploit **data-level parallelism**, even if the overhead is likely to be higher than in a SIMD computer

**Tightly coupled** MIMD architectures exploit **thread-level parallelism** since multiple cooperating threads operate in parallel

**Loosely coupled** MIMD architectures (cluster and warehouse-scale computers) exploit **request-level parallelism**, where many independent tasks can proceed in parallel naturally with little need for communication or synchronization



# High performance computers

- 1961 **IBM 7030 Stretch** →  $10^6$  Flops/sec (**megaFLOPS** or **MFLOPS**)
  - scalar processors
- 1984 **M-13** →  $10^9$  Flops/sec (**gigaFLOPS** or **GFLOPS**)
  - vector processors, shared memory
- 1997 **ASCI Red** →  $10^{12}$  Flops/sec (**teraFLOPS** or **TFLOPS**)
  - massive parallelism, distributed systems, message passing
- 2008 **IBM Roadrunner Red** →  $10^{15}$  Flops/sec (**petaFLOPS** or **PFLOPS**)
  - multicore processors, precision extension, fault tolerance

# High performance computers

- 2011 **Fujitsu K** → 10,5 petaFLOPS
- 2016 **Sunway TaihuLight** → 93 petaFLOPS
- 2018 **Summit** → 122 petaFLOPS
- 2020 **Supercomputer Fugaku** → 415 petaFLOPS
- 2022 **Frontier** →  $10^{18}$  Flops/sec (exaFLOPS or EFLOPS)

See <https://www.top500.org/>