ADVANCED ARCHITECTURE

Annalisa Massini

2025-2026

Lectures 1 and 2

COURSE INFORMATION

Course topics

Three years ago, I started to change the focus of this course and I also changed its name.
At the moment, the course is more focused on architectures than on applications and notions of quantum computing are also introduced

The course will focus on:

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

Time, venue and course page

- **Tuesday** 14:00-17:00 - Aula G50 (Edificio G) - Viale Regina Elena 295
- **Thursday** 11:00-13:00 - Aula 2L - Via del Castro Laurenziano 7a

- Lectures will be given mainly by using slides
- Exercises will be done 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 and access the course material
- Register to course **2kiiwkid** on classroom using your Sapienza email (surname.matricola@studenti.uniroma1.it) to receive notices and additional material

Exam

Exam will consist of:

- Written part
  - Two partial exams (*Midterm + end-of-term*) **or** a final exam – *consisting of a written test with exercises.*
- Oral part
  - Presentation of one or more papers ***or*** Development and presentation of a project ***or*** Oral exam
  - *NOTE that project topic and papers for presentation must cover* one course topic *and* ***must be approved by the teacher***

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
  5th Ed. 2011 and 6th 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 IntegratedApproach
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 of the Von Neumann architecture
- Circuits for arithmetic operations and circuit evaluation
- Number representations for Fast Arithmetic
- Motivation to parallel architectures and their classifications
- Sparse matrices: compact storage methods

Course topics

Part 2 – Parallel architectures and quantum circuits

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

INTRODUCTION

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

Introduction

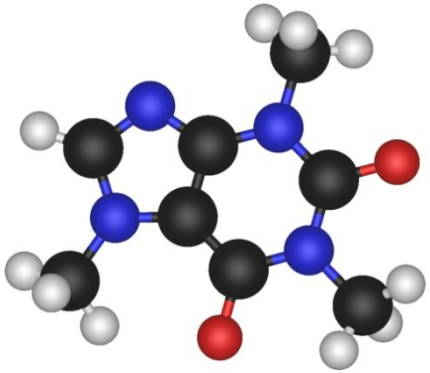
- Traditional **methods in science** and **engineering** are:
  - To develop *theories* and *projects*
  - To execute *experiments* and
  - To build *systems*

- 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

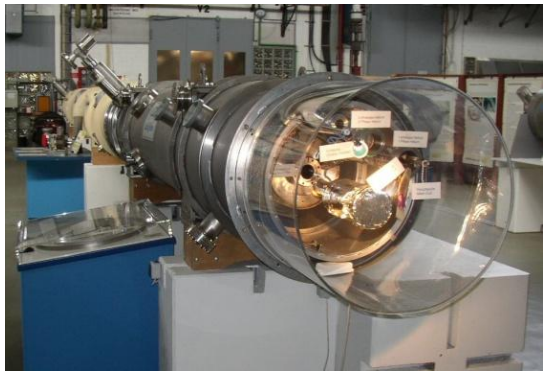
- **Computers** represent the fundamental tool for the simulation and can be seen both as a **microscope** and as a **telescope** with respect to space and to the time



- **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

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

$$\frac{\mathrm{dLux}}{\mathrm{dt}} = \left(\frac{K_2}{1 + A * Ls1^2}\right) * N - (d_2 + r) * Lux + \delta_2$$

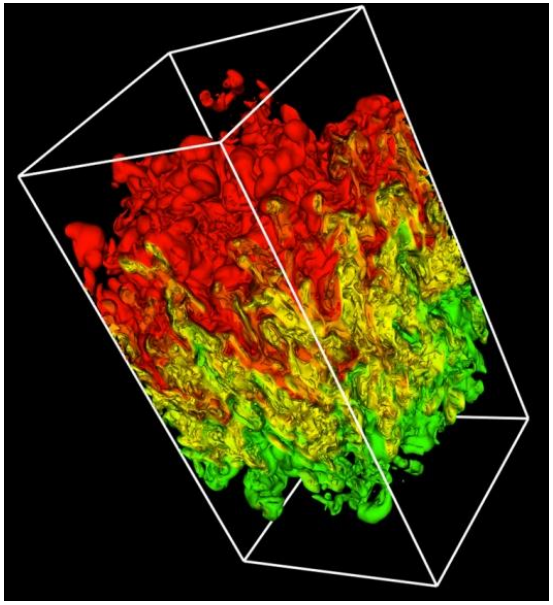
$$\mathrm{Z} = Z_1 + Z_2$$

$$Z_1 = K_3 * Lux$$

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

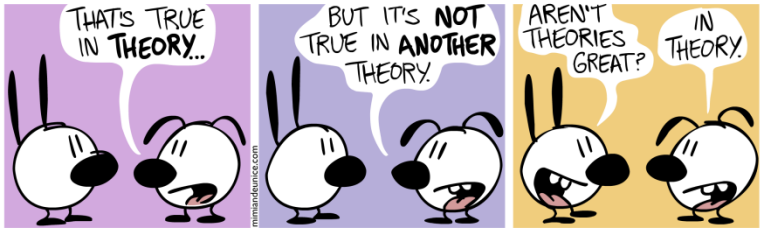
$$\frac{\mathrm{dLs1}}{\mathrm{dt}} = Ls * \left(1 - \frac{Ls}{K5}\right) * Z * \left(1 - \frac{Z}{K6}\right) \qquad (1)$$

- Computational science involves:
  - The application of **computer** simulation
  - Different forms of computation (numerical analysis, theoretical computer science, etc.)
  - Problems in various scientific disciplines



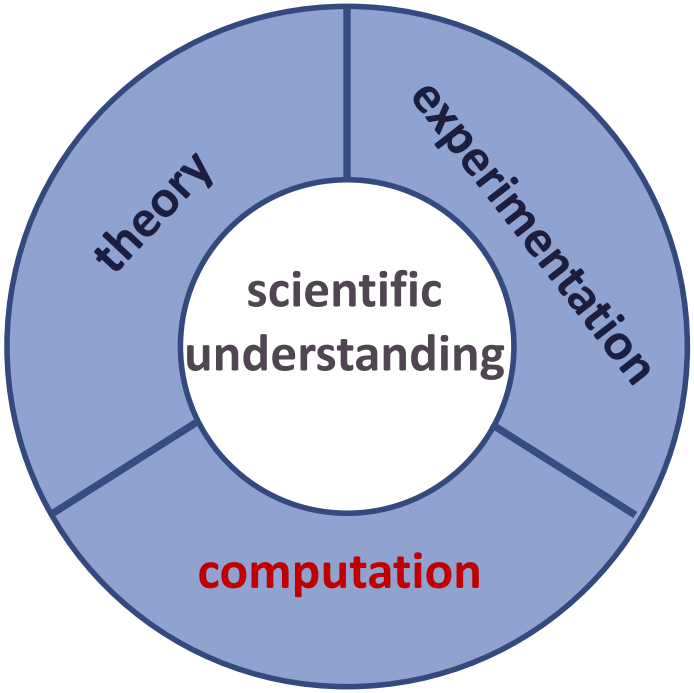
Computational Science

- 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, 2019

Chapter 1

Computer Technology Progress

- Computer technology has made incredible progress since the first general-purpose electronic computer was created, about 70 years
  - **Performance Comparison**: A modern cell phone (<$500) matches the performance of a 1993 supercomputer ($50 million)
  - **Two Growth Drivers**: Improvement stems from both advances in semiconductor technology and innovations in computer design (architecture)
  - **Architectural Progress**: While technology improved steadily, architectural progress was less consistent until the microprocessor era

The Emergence of the Microprocessor

- **Early Growth (1945–1970s):** Performance improved by ~25% per year through a balance of technology and architecture
- **Late 1970s**: The microprocessor allowed designers to ride the wave of Integrated Circuit (IC) improvements directly

- **The Microprocessor Jump**: The shift to microprocessors increased growth to ~35% per year by leveraging integrated circuit (IC) improvements
- **Economic Impact and Market Shift**: Mass production led to cost advantages, causing the **microprocessor** to dominate the computer industry

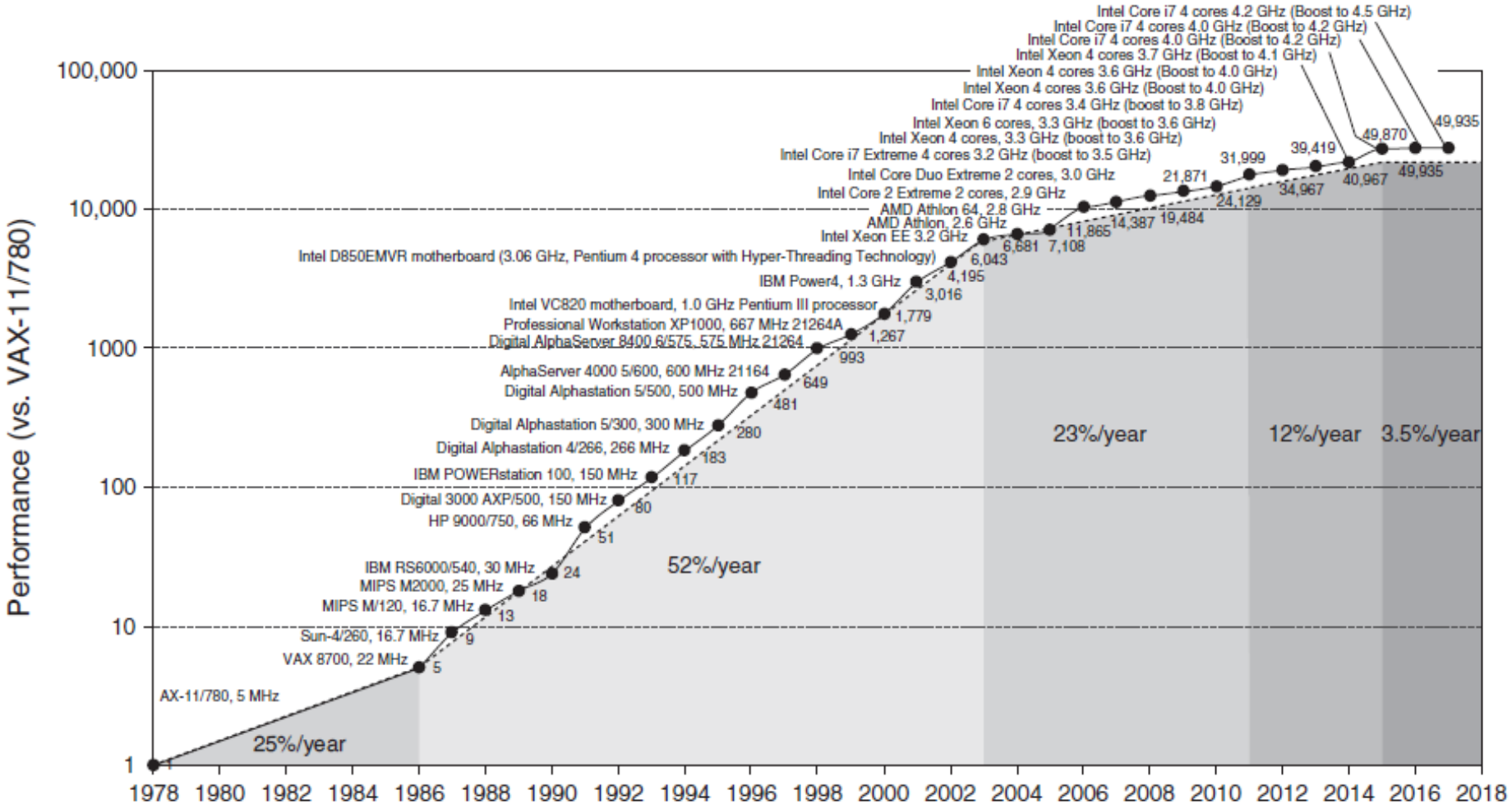
The Pillars of the RISC Revolution

- New architectures – **RISC (Reduced Instruction Set Computer)** – succeeded in the 1980s thanks to:
  - **Software Independence**: The virtual elimination of assembly language programming reduced the need for binary (object-code) compatibility
  - **Standardized Systems**: The rise of vendor-independent operating systems like UNIX and Linux lowered the risk of adopting new, innovative designs
  - These factors allowed architects to **focus on performance** rather than legacy compatibility

Golden Era of Performance (1986–2003)

- **RISC-based computers** raised the performance bar, forcing prior architectures to keep up or disappear
  - **Technological Drivers**: RISC designs enabled the effective use of:
    - Instruction-Level Parallelism (ILP),utilizing techniques like pipelining and through multiple instructions
    - Cache memory
  - **Annual Growth**: Performance increased by 52% per year, meaning it doubled every 1.5 years, during this *Renaissance* of computer design

The Emergence of the Microprocessor



This figure shows that the combination of architectural and organizational enhancements led to 17 years of sustained growth in performance at an annual rate of over 50%

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 supercomputers 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 were replaced by ***servers*** made using ***microprocessors***
    - ***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**
  - Quantitative Success (by 2003): High-performance microprocessors became 7.5 times faster than what technology improvements (like circuit design) alone would have allowed
  - **Architecture + Technology** gave a **52%** performance growth per year vs with technology alone

Computer architectures evolution

- **Fourth impact**
  - The hardware renaissance had impact on **software development**
    - Instead of performance-oriented languages like C and C++, much more programming today is done in managed programming languages like Java, C# and Scala
    - 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**
- The physical limit ended the era of faster clock speeds.
- In 1974 Dennard observed that the power density remained constant as transistor size shrank, that is they could run faster while using less power
- In 2004 the scaling ended because voltage and current could no longer drop without compromising circuit reliability
- As an industrial impact, in 2004, Intel and others canceled high-performance uniprocessor projects, declaring that the road to higher performance would be via ***multiple processors per chip*** *rather than via faster uniprocessors*

Computer architectures evolution

- There was a shift from implicit hardware optimization to explicit software parallelism
  - **Implicit Parallelism (ILP):** Instruction-Level Parallelism was exploited by compilers and hardware without requiring the programmer's attention.
  - **Explicit Parallelism:** To gain performance now, applications must be restructured for:
    - **DLP:** Data-Level Parallelism.
    - **TLP:** Thread-Level Parallelism.
    - **RLP:** Request-Level Parallelism found in Warehouse-Scale Computers
- This shift requires **programmers** to undergo the major effort of **restructuring applications** to exploit explicit parallelism

Computer architectures evolution

Theoretical and Physical Limits - Amdahl and Moore

- **Amdahl's Law:**
  - Defines the practical limit of parallel speedup
  - If 10% of a task is serial, the maximum speedup is 10x, regardless of the number of cores added
- **Moore's Law:**
  - In 1965 Gordon Moore predicted that transistor counts would double every two years
  - That prediction lasted for about 50 years, but no longer holds
- **Reality Check (2010 vs. 2016):**
  - **Moore's Prediction:** Expected ~18.7 billion transistors by 2016
  - **Actual Result:** Intel microprocessors reached only ~1.75 billion—a factor of 10 difference from the prediction

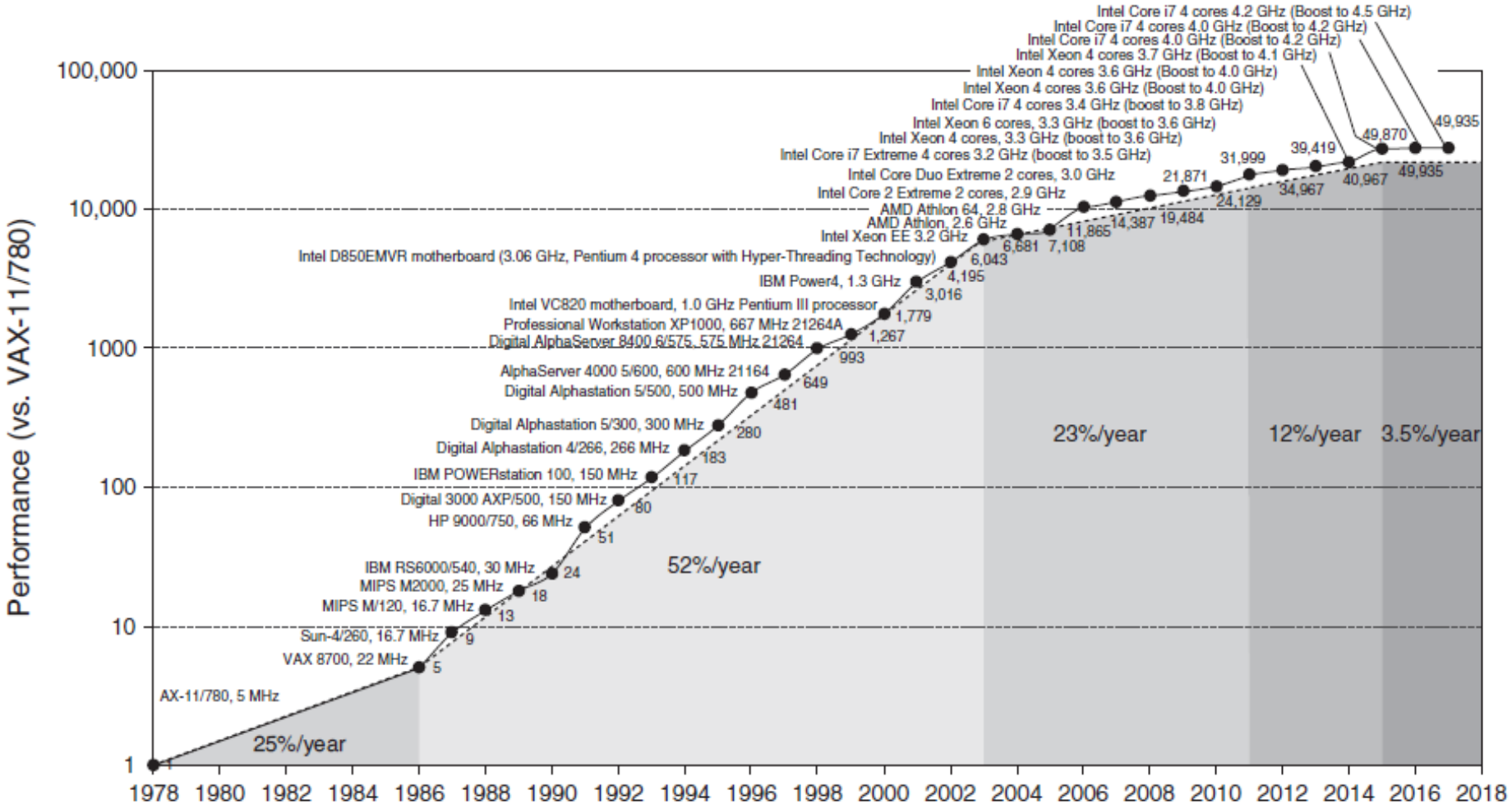
Computer architectures evolution

Combination of factors that caused improvements in processor performance to **slow down**:

- Transistors are no longer getting significantly better or denser
- Power density limits prevent further increases in clock speed
- Microprocessors must operate within strict thermal and energy limits
- Amdahl's Law limits the effectiveness of adding more cores

- **The impact**: Processor performance now doubles roughly every 20 years, a massive drop from the 1.5-year doubling rate seen during the Golden Era (1986–2003)

The Emergence of the Microprocessor



This figure shows that the combination of architectural and organizational enhancements led to 17 years of sustained growth in performance at an annual rate of over 50%

GENERATION OF COMPUTERS AND CLASSES

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**

Classes of Computers

- Current changes are redefining how computers appear, how they are used, and how the market is structured
- This evolution has resulted in **five distinct computing sectors**
  - **Personal mobile device** (PMD)
  - **Desktop computing**
  - **Servers**
  - **Clusters**
  - **Embedded computers**
- Each market is defined by its own:
  - **Specific Applications**
  - **Unique Requirements**
  - **Tailored Computing Technologies**

Internet of Things/Embedded Computers

- **Embedded computers** are are integrated into everyday machines like microwaves, washing machines, networking switches, and automobiles
- **The IoT Concept** refers to embedded computers connected to the Internet (typically wirelessly)
- **Smart Ecosystems** are obtained when IoT devices, combined with sensors and actuators, interact with the physical world, enabling:
  - **Personal**: Smart watches, speakers, and thermostats
  - **Large Scale**: Smart cars, homes, grids, and cities

- As a scale of growth: in 2020 it was estimated between 20 to 50 billion connected IoT devices.

Internet of Things/Embedded Computers

Design Constraints of Embedded Computing

- **Wide diversity in power and cost:**
  - 8-bit/32-bit processors costing one penny
  - high-end 64-bit processors (e.g., in cars) costing $100

- **Price** is the dominant factor
- **Performance** requirements do exist, but the primary goal is often meeting the **performance** need at a **minimum price**, rather than achieving higher performance (at a higher price)

Personal Mobile Device

- **Personal mobile device** (PMD) is the term used for 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 battery use, the requirement for inexpensive packaging (plastic vs. ceramic), and the absence of cooling fans
- Energy and size requirements lead to use of Flash memory for storage instead of magnetic disks

Personal Mobile Device

- **PMDs** are distinguished from other embedded devices by their ability to run third-party (externally developed) software
- **Responsiveness** is critical for media-oriented applications (video, audio, web-based tasks)
- **Real-Time Requirements** are highly application-dependent**:**
  - **Hard Real-Time:** Absolute maximum execution time (e.g., frame processing in video)
  - **Soft Real-Time:** Occasional misses of time constraints are acceptable as long as the average performance remains high
- **Resource Optimization**
  - **Code Size:** Because memory represents a significant portion of system cost, designers emphasize compact code size
  - **Energy Efficiency** driven by capacity of batteries and limits of heat dissipation

Desktop Computing

- Historically the desktop computing market is the largest market in dollar terms, though sales are currently declining
- **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 newest and most powerful microprocessors, as well as the most efficient cost-reduced chips, debut in the desktop market

Desktop Computing

- The **desktop market** tends to be driven to optimize *price-performance*
- **Performance** success is measured primarily by:
  - **Compute performance** corresponding to general processing speed
  - **Graphics performance** crucial for visual and professional applications
- **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
- The rise of web-centric and interactive applications is creating new complexities for traditional performance evaluation

Servers

- 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

Servers

- 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
  - They need the ability to expand computing capacity, memory, storage, and I/O bandwidth in response to increasing demand or functional growth
- Servers are designed for **efficient throughput**
  - Success is measured by **overall efficiency** rather than just individual responsiveness.
  - **Key Metrics:** Transactions per minute or web pages served per second
  - **Cost-Effectiveness:** Designers focus on the number of requests handled per unit of time to ensure economic viability for large enterprises

Clusters/Warehouse-Scale Computers

- The growth of the class of computers called **clusters** is due to the growth of Software as a Service (SaaS) for many applications: search, social media, video sharing, online shopping, etc.

- **Clusters** are:
  - Collections of desktop computers or servers connected by *local area networks* to act as a single larger computer
  - Each node runs its own operating system, and nodes communicate using a networking protocol
- **WSCs** (Warehouse-Scale Computers) are
  - The largest form of **clusters**—collections of tens of thousands of servers to act as a single, massive computer

Clusters/Warehouse-Scale Computers

- Price-performance and power are critical to WSCs since they are so large, and the **majority of the cost** of a warehouse is associated with **power and cooling** of the computers inside the warehouse
- While servers use expensive hardware, WSCs use **redundant, inexpensive components** and rely on a software layer to isolate frequent hardware failures

- **WSCs** and **supercomputers** are equally expensive, but:
  - **Supercomputers** focus on floating-point performance and long-running (weeks), communication-intensive batch programs
  - **WSCs** focus interactive applications, large-scale storage, dependability, and high Internet bandwidth

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

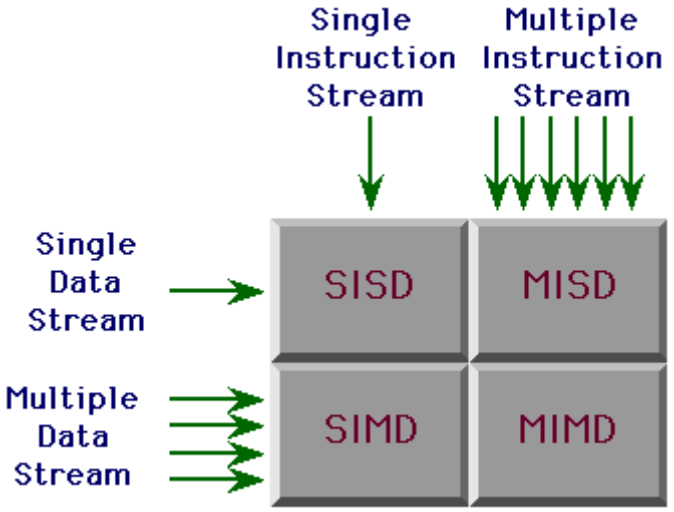
- **Michael Flynn** studied the parallel computing efforts in the 60s, and introduced a **taxonomy** of computer architectures that is still the *most common way of categorizing* systems defining abbreviations we still use today

- 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

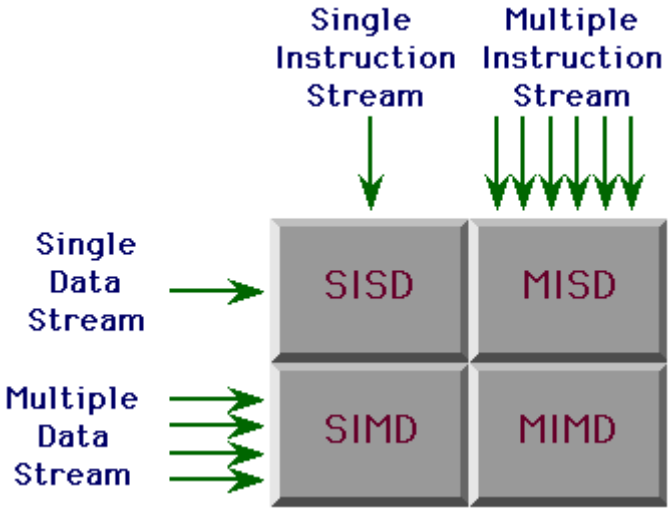
  - Single Instruction, Single Data - **SISD**
  - Single Instruction, Multiple Data - **SIMD**
  - Multiple Instruction, Single Data - **MISD**
  - Multiple Instruction, Multiple Data - **MIMD**



Taxonomy of Computer Architectures

Single Instruction stream, Single Data stream – SISD

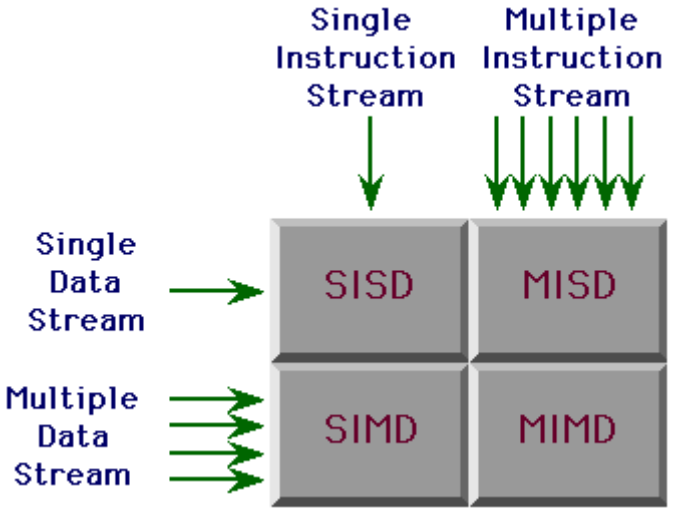
- This category is the **uniprocessor**
- The programmer thinks of it as the standard sequential computer, but it can exploit instruction-level parallelism
- SISD architectures use ILP techniques such as superscalar and speculative execution



Taxonomy of Computer Architectures

Single Instruction stream, Multiple Data stream - SIMD

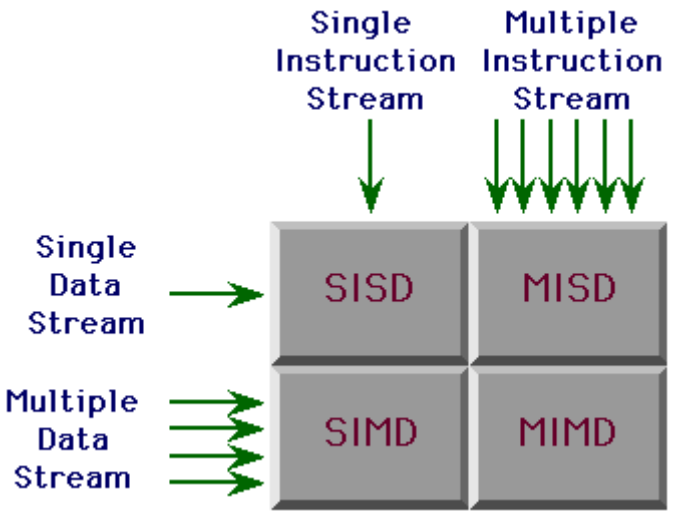
- The same instruction is executed by multiple processors using different data streams
- SIMD computers exploit data-level parallelism by applying the same operations to multiple items of data in parallel
- Each processor has its own data memory, but there is a single instruction memory and control processor, 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

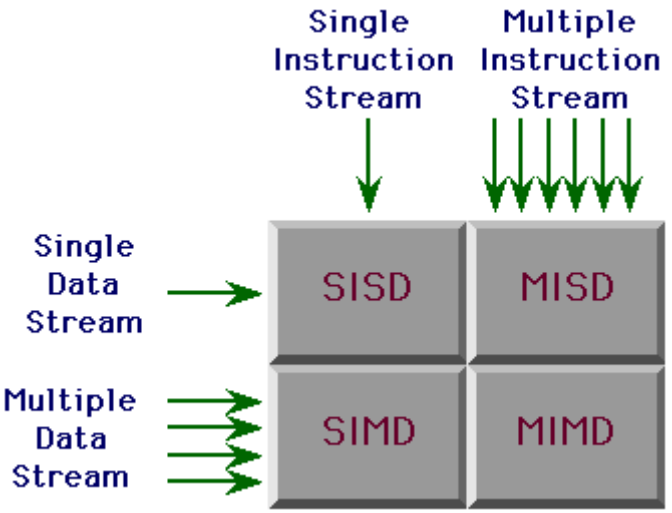


Taxonomy of Computer Architectures

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 - independent tasks can proceed in parallel with little need for communication or synchronization



Defining Computer Architecture

- The task of a **computer designer** is complex: *balancing performance and energy efficiency against strict constraints in cost, power, and availability*

- **Broad Scope of Design**: Instruction Set Architecture (ISA).
  - Functional organization and logic design
  - Physical implementation, including IC design, packaging, power, and cooling

Defining Computer Architecture

- **Cross-Disciplinary Optimization**: Success requires deep familiarity with everything from compilers and OS to hardware
- According to the past, ***architecture*** meant only Instruction Set Design and other tasks were ***just implementation***
- In a modern view, the technical hurdles in organization and implementation are often **more challenging than the ISA itself**

Defining Computer Architecture

- The **Instruction Set Architecture (ISA)** serves as the critical boundary between software and hardware, representing the programmer-visible instruction set
- An ISA is defined by **seven key dimensions**

- The most popular RISC processor is **ARM** (Advanced RISC Machine)
- ARM is the leader in volume, with 14.8 billion chips shipped in 2015, 50x the volume of 80x86 (the traditional standard for desktops and servers)
- **Comparative Study:** Modern analysis typically focuses on **80x86**, **ARMv8**, and **RISC-V**

Defining Computer Architecture

- **RISC-V** is a modern RISC instruction set developed at UC Berkeley
  - RISC-V is a free and elegant example of RISC architecture and openly adoptable ISA
  - Designed 30 years after the first RISC sets
- It incorporates best practices while avoiding legacy mistakes:
  - **Large register set**
  - **Lean operation set**
  - **Easy-to-pipeline instruction format**
- Supported by over 60 major companies (including: AMD, Google, Nvidia, Microsoft, Samsung)
- **The RISC-V integer core** serves as the primary example ISA for modern **quantitative analysis**

The seven dimensions of an ISA

1 - Classes of Instruction Set Architecture (ISA)

- The standard for modern ISAs is **General-Purpose Register Architectures**, where operands are stored in either registers or memory locations
  - **80x86:** 16 General-Purpose (GP) and 16 Floating-Point (FP) registers.
  - **RISC-V:** 32 General-Purpose (GP) and 32 Floating-Point (FP) registers.
- Two Major Sub-Classes**:**
  - **Register-Memory ISAs (e.g., 80x86):** Can access memory directly as part of many different instructions
  - **Load-Store ISAs (e.g., RISC-V, ARMv8):** Memory access is restricted exclusively to specific *load* and *store* instructions
- Every ISA announced since 1985 has followed the **Load-Store** architecture

The seven dimensions of an ISA

2 - Memory Addressing

- Virtually all modern desktop and server computers, including the 80x86, ARMv8, and RISC-V, use **byte addressing** to access **memory operands**
- Some architectures require object to be **aligned**:
  - An access to an object of size *s* bytes at byte address *A* is aligned if: *A* mod *s* = 0
  - **ARMv8**: Requires that objects must be aligned
  - **80x86** and **RISC-V**: Do not require alignment for memory access
- Performance Impact: accesses are generally faster if operands are aligned (even if it is not mandatory for all)

The seven dimensions of an ISA

3 - Addressing Modes

- Addressing modes specify the address of a memory object in addition to identifying registers and constant operands

- **RISC-V** focuses on simplicity with three modes:
  - **Register, Immediate** (for constants), **Displacement** (Register + Constant offset)

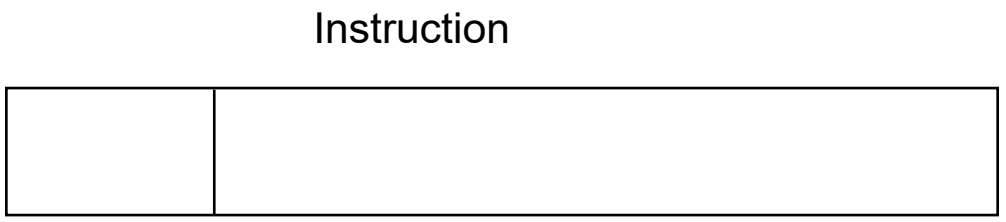
The seven dimensions of an ISA

3 - Addressing Modes

- **80x86** supports the RISC-V modes plus variations:
  - **Absolute** (no register), **Based Indexed with Displacement** (two registers + displacement), **Based with Scaled Index** (one register multiplied by operand size), Additional modes: Register indirect, indexed, and based with scaled index
- **ARMv8** includes RISC-V modes plus:
  - **PC-relative** addressing, **Sum of two registers** (with or without scaling), **Autoincrement/Autodecrement:** The calculated address automatically replaces the contents of the base register

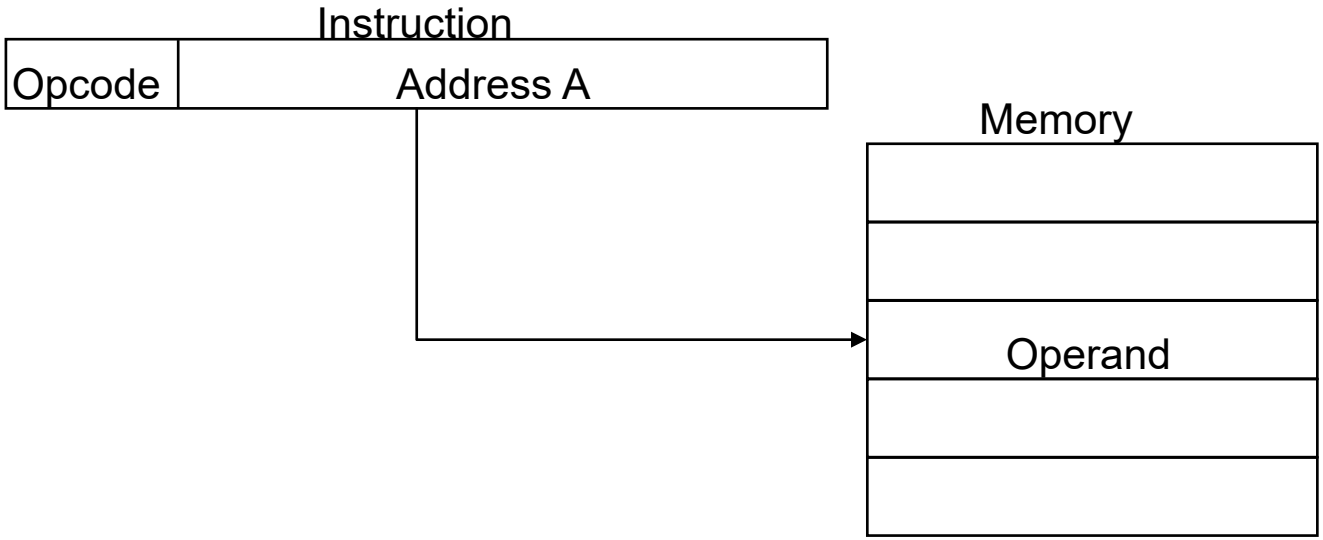
Immediate Addressing

- **Operand is part of instruction**
- Operand = address field
- e.g. ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference to fetch data
- Fast
- Limited range



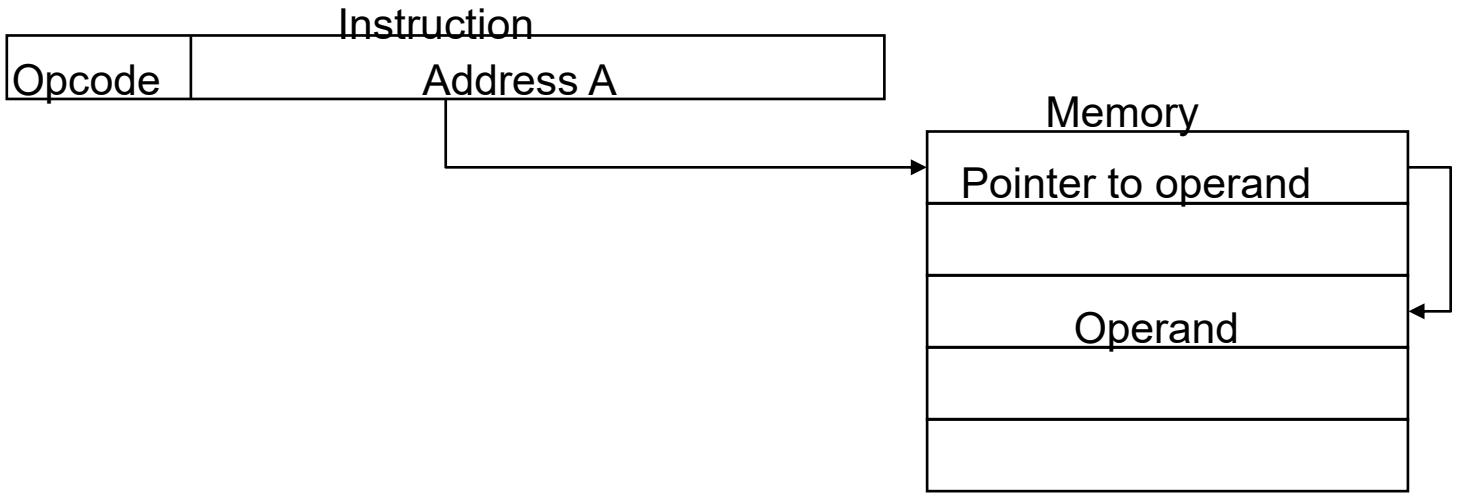
Direct Addressing

- Address field contains **address of operand**
- Effective address (EA) = address field (A)
- Single memory reference to access data
- No additional calculations to work out effective address
- Limited address space



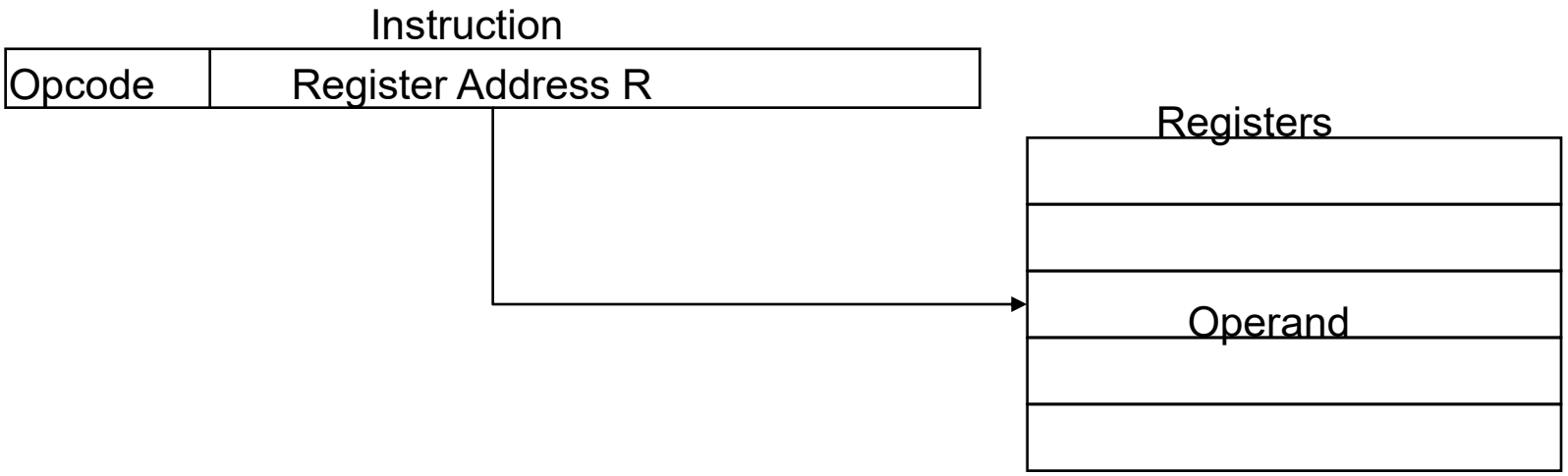
Indirect Addressing

- **Memory cell** pointed to by address field contains the address of (pointer to) the operand
- EA = (A)
  - Look in A, find address (A) and look there for operand
- Large address space - 2^n where n = word length
- Multiple memory accesses to find operand - hence slower



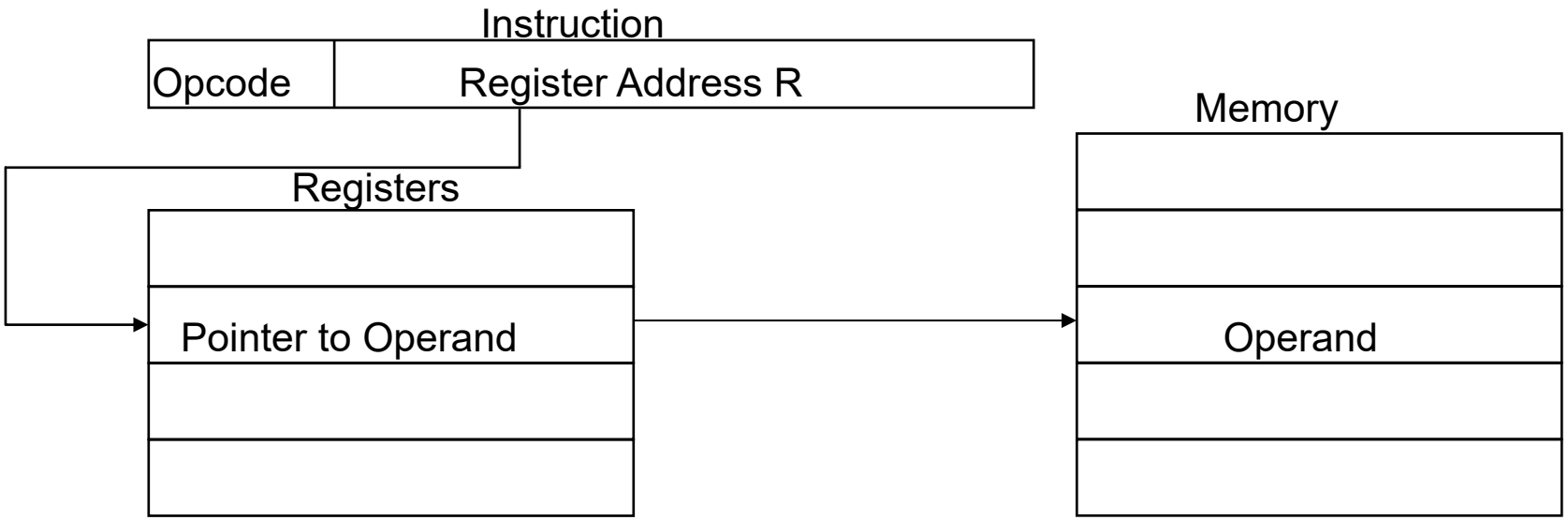
Register Addressing

- Operand is held in **register** named in address filed
- Limited number of registers
- Very small address field needed
  - Shorter instructions - Faster instruction fetch
- No memory access -Very fast execution
- Very limited address space
- Multiple registers helps performance



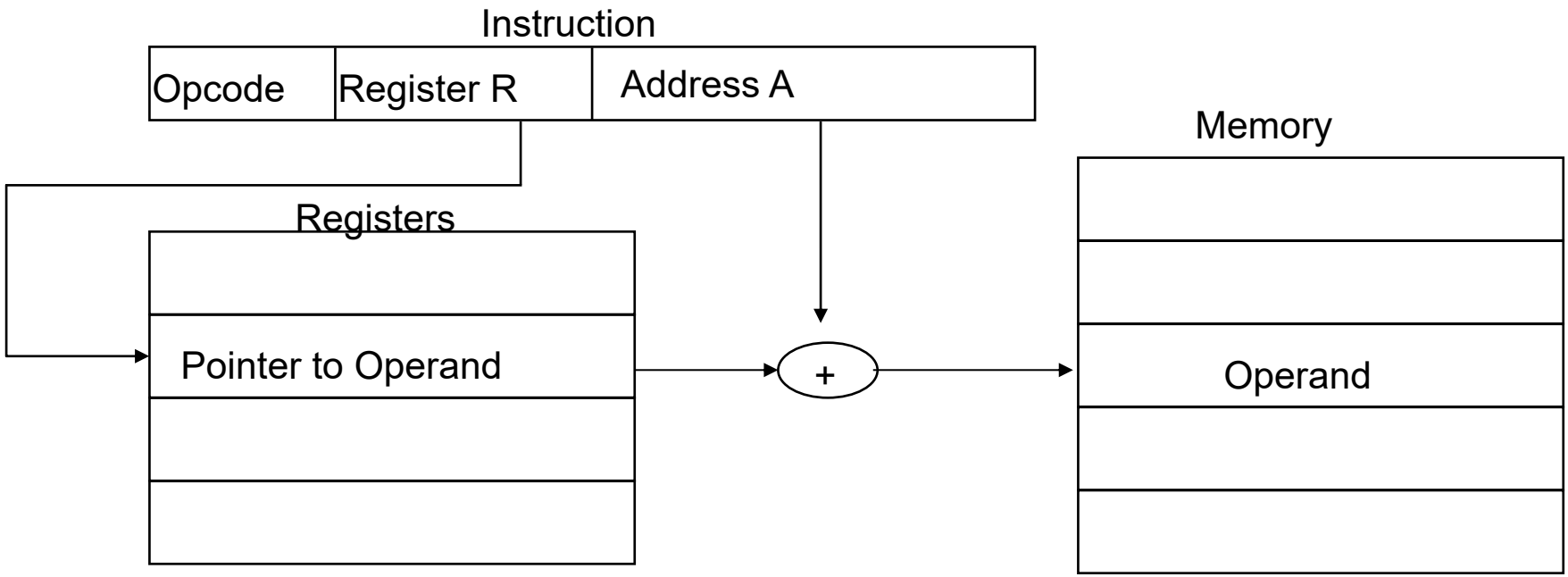
Register Indirect Addressing

- It is **indirect addressing**
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- Large address space (2^n)
- One fewer memory access than indirect addressing



Displacement Addressing

- EA = A + (R)
- Address field hold **two values**
  - A = base value
  - R = register that holds displacement
  - or vice versa



Relative Addressing

- A version of **displacement addressing**
- R = Program counter, PC
- EA = A + (PC)
- i.e. operand from A cells from current location pointed to by PC
- locality of reference & cache usage

Base-Register Addressing

- A version of **displacement addressing**
- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit
- e.g. segment registers in 80x86

Indexed Addressing

- A version of **displacement addressing**
- A = base address
- R = displacement
- EA = A + R
- Good for accessing arrays
  - EA = A + R
  - R++

Stack Addressing

- A **stack** is a linear array of reserved memory locations
- Can be sometimes referred to as a ***pushdown list*** or ***last-in-first-out queue***
- Items are appended to the top of the stack
- At any given time, the location block is partially filled
- Associated with the stack is a pointer - **stack pointer** - whose value is the address of the top of the stack

The seven dimensions of an ISA

4 - Types and Sizes of Operands

- Modern ISAs (80x86, ARMv8, and RISC-V) provide standardized support **for various data widths**:
  - 8-bit: ASCII characters
  - 16-bit: Unicode characters or Half Words
  - 32-bit: Integers or Words
  - 64-bit: Long integers or Double Words
- Full support for **Floating-Point Standards** - IEEE 754 standard:
  - 32-bit: Single Precision
  - 64-bit: Double Precision
- The 80x86 includes unique support for 80-bit Floating Point (Extended Double Precision)

The seven dimensions of an ISA

5 - Operations

- Operations across modern architectures are divided into four primary functional groups:
  - **Data Transfer:** Moving data between registers and memory
  - **Arithmetic Logical:** Standard operations (Add, Sub,AND, OR)
  - **Control:** Instructions that change the flow of program execution
  - **Floating Point:** Dedicated operations for real-number calculations

- RISC-V is a simple and easy-to-pipeline ISA, highly representative of the modern RISC architectures utilized today
- The 80x86 possesses a much richer and larger set of operations, characteristic of CISC (Complex Instruction Set Computer) designs

The seven dimensions of an ISA

6 - Control Flow Instructions

- Virtually all ISAs (80x86, ARMv8, and RISC-V) provide basic control mechanisms:
  - Conditional branches
  - Unconditional jumps
  - Procedure calls and returns
- All three architectures utilize PC-relative addressing, where the target address is calculated by adding an offset field to the PC
- Branching Differences:
  - RISC-V: Conditional branches (e.g., BEQ, BNE) directly test the contents of registers
  - 80x86 and ARMv8: Branches test condition code bits (flags) set as side effects of prior arithmetic/logic operations

The seven dimensions of an ISA

7 – Encoding an ISA

- There are **two basic choices**:
  - Fixed Length: Simplifies instruction decoding hardware
  - Variable Length: Can reduce overall program size
- Architectural Implementations:
  - RISC-V & ARMv8: Use 32-bit fixed-length instructions
  - 80x86: Uses variable-length encoding, ranging from 1 to 18 bytes
- **Design Trade-offs:**
  - **Code Size:** Programs for 80x86 are usually smaller than standard 32-bit RISC-V programs because instructions only use the space they need
  - **Complexity Factors:** The number of registers and addressing modes significantly impact instruction size, as these fields may appear multiple times

Organization and Hardware

- The Two Components of Implementation:
  - **Organization** (also Microarchitecture): High-level design aspects including the memory system, interconnects, and internal CPU design (arithmetic, logic, branching, data transfer)
  - **Hardware**: Detailed logic design and packaging technology

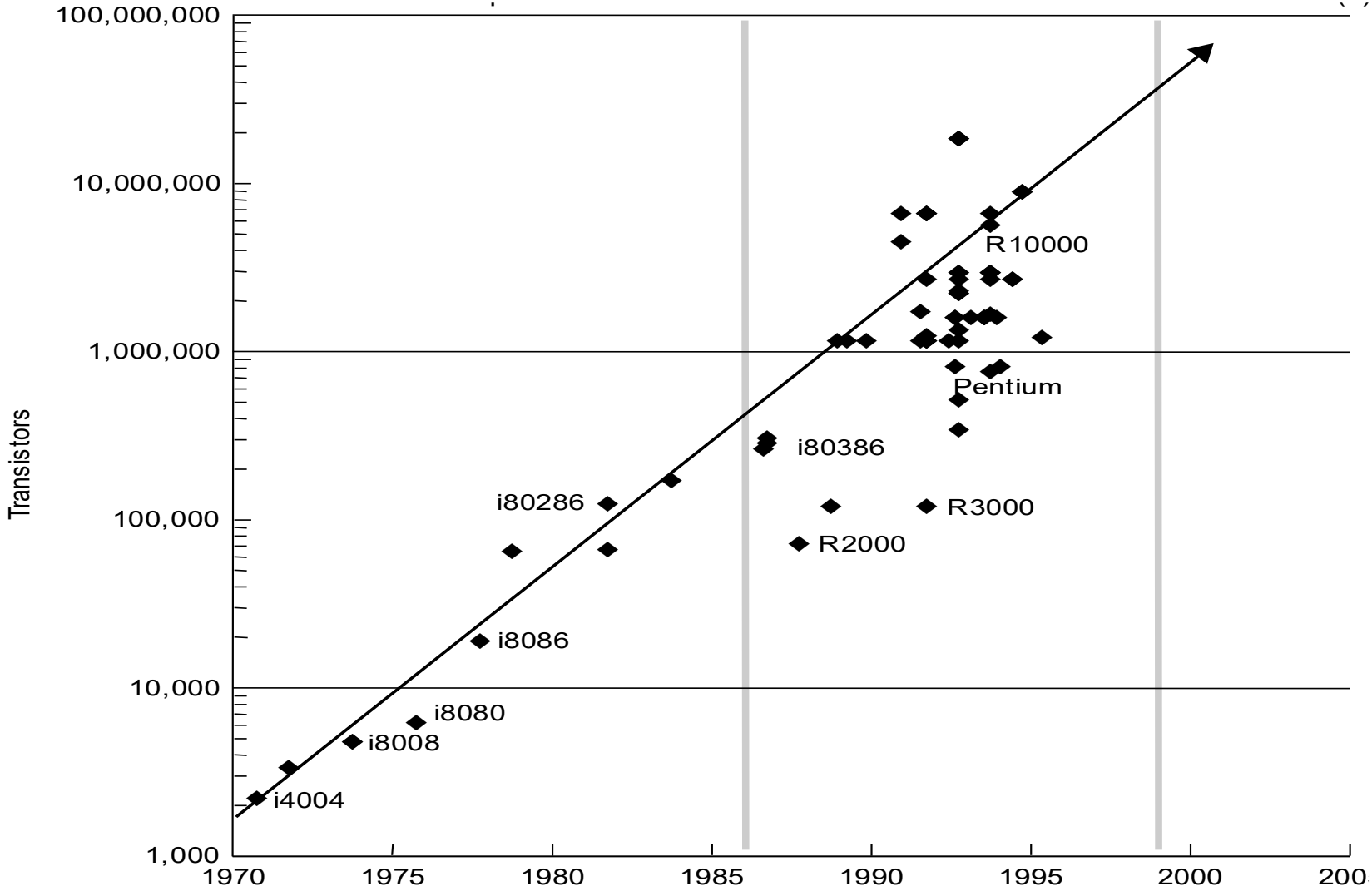
- **Key Distinction:**
  - Processors like the **AMD Opteron** and **Intel Core i7** share the same ISA (80x86) but have different **organizations** (pipelines and caches)
  - The **Intel Core i7** and **Xeon E7** may have similar organizations but differ in **hardware** (clock rates and memory systems) to target different markets (Desktop vs. Server)

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

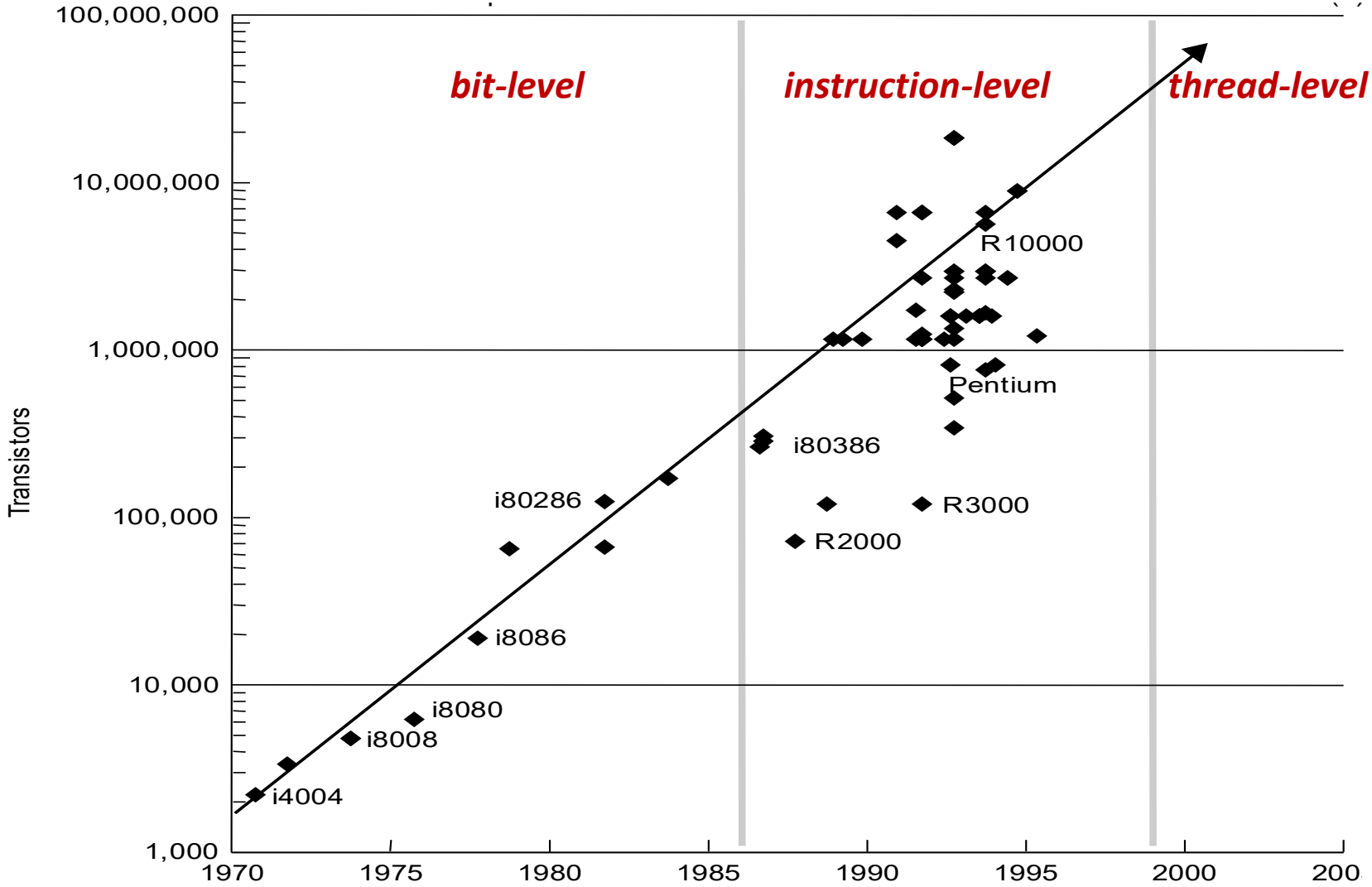
Architectural Trends

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



Architectural Trends

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



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**
- 2023 **Frontier** → **10^{18}** Flops/sec (**exaFLOPS** *or* **EFLOPS**)

See https://www.top500.org/