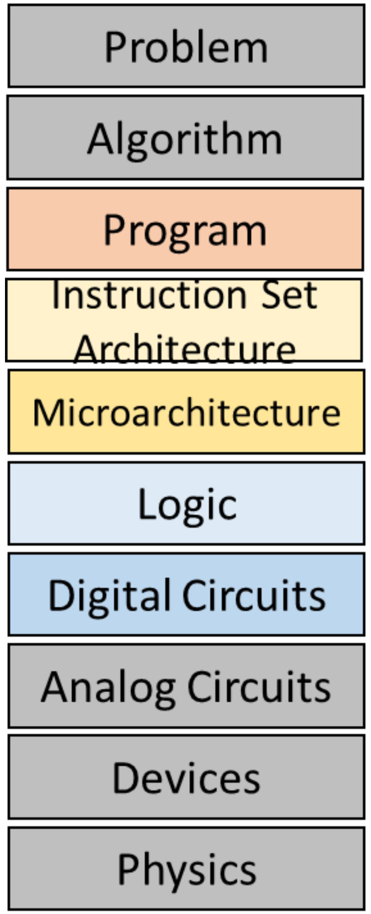
An Introduction to Computer Architecture

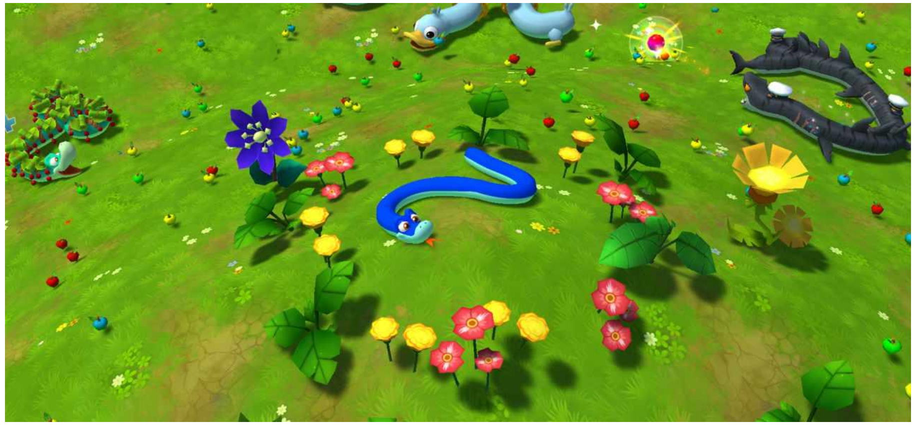
Prof. Dr. Rolf Drechsler
Dr. Muhammad Hassan
M.Sc. Jan Zielasko
M.Sc. Milan Funck



We Have Come so Far!

[1] licensed under CC BY-ND

[2] licensed under CC BY-NC



[3] licensed under CC BY-NC

Snake game on Nokia 3310
Circa **2000**

Snake game on Iphone
Circa **2022**

Performance

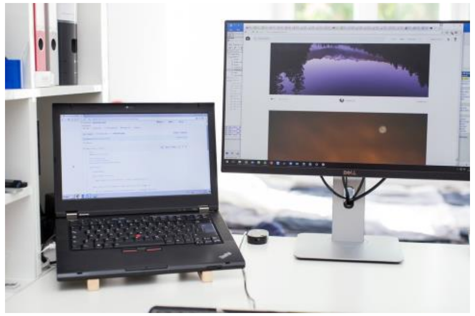
1. https://www.comptoir-hardware.com/actus/business/26786-ouf-il-y-aura-encore-des-telephones-nokia-finalement-.html
2. https://www.iphonemod.net/wp-content/uploads/2019/09/iphone-11-pro-rainbow.jpg
3. https://www.iphonemod.net/snake-rivals-io-snakes-game.html

and future seems ….

ChatGPT

This Course

- Aims to introduce the key concepts and ideas in computer architecture
- Explores the design of modern microprocessors
- Examines important trends and current and future challenges

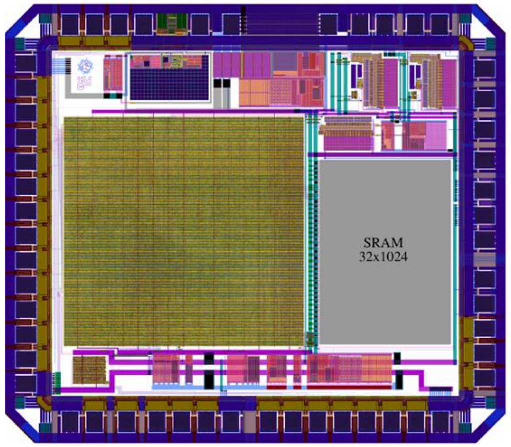


Lecture Outline

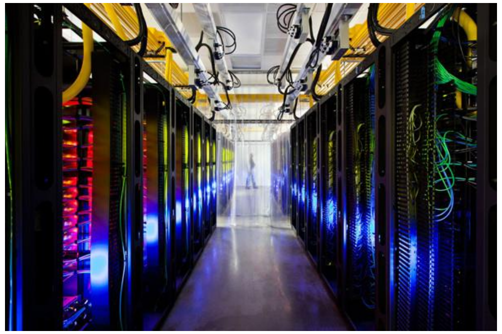
- What is computer architecture?
- Important topics in computer architectures
- Computer architecture design goals
- From high-level language to the hardware language
- Main components in a computer
- CPU architecture
- Verification and testing
- Future trends with multicore processors, systems on chip (SoCs), and beyond

Introduction

- The modern microprocessor is often the engine behind how we communicate and work.
- It has helped to create our digital world where communication, computation, and storage is almost free.
- It underpins many scientific breakthroughs and can help us make better use of the world's limited resources



RISC-V processor core[1]



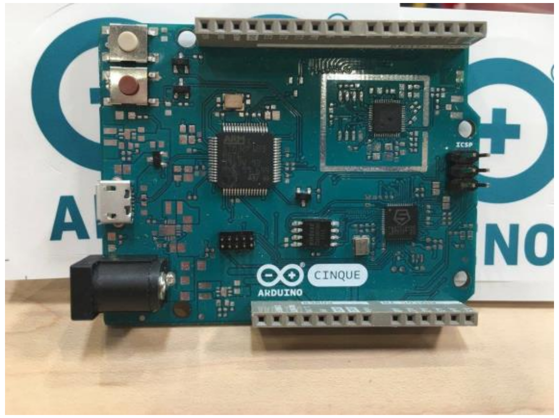
Google Data Center
By Connie Zhou, CC BY-NC 4.0

[1] https://www.electronics-lab.com/x-fab-efabless-announce-successful-first-silicon-raven-open-source-risc-v-microcontroller/ (Licensed under CC BY-SA)

Introduction

- The modern computer is less than 100 years old.
- The first electromechanical and valve-based machines were produced in the 1930s and 1940s.
- Today’s machines are many orders of magnitude faster, lower power, more reliable, and cheaper.

EDSAC replica (2018)[1]

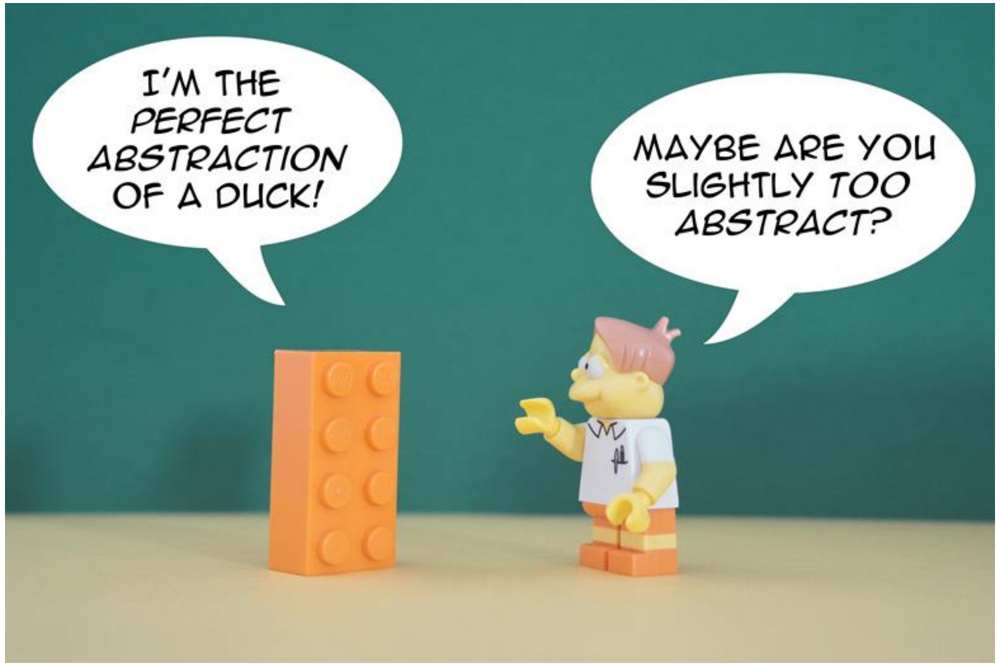


Arduino Cinque with RISC-V core[2]

[1] EDSAC photo, CC BY-SA 4.0

[2] https://www.electronics-lab.com/cinque-combining-risc-v-arduino/ (Licensed under CC BY-SA)

Abstractions


I'M THE PERFECT ABSTRACTION OF A DUCK!
MAYBE ARE YOU SLIGHTLY TOO ABSTRACT?

Abstraction is attractive, sexy, old and modern at the same time.

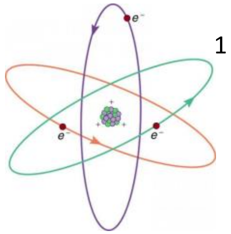
We think of it as something very loosely related to reality, and it often conveys the sense of something *difficult to understand*

A bunch of buttons and knobs in order to wash whatever you want to wash.

To wash your clothes, do you need to know how the washing machine works internally?

https://thevaluable.dev/abstraction-type-software-example/

What is a Computer System?

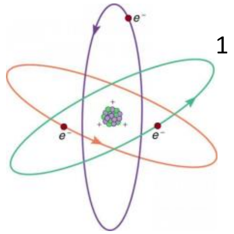


[1]

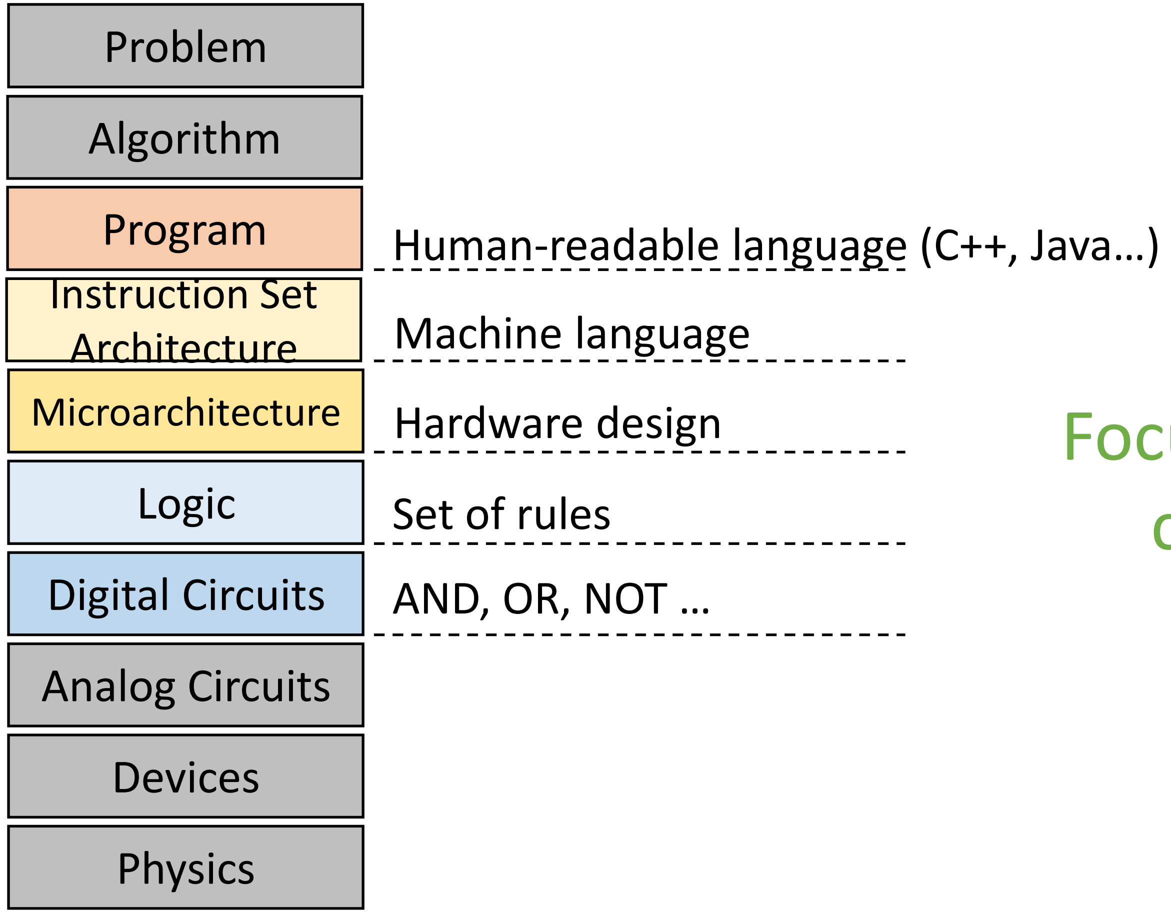
Problem	Who scores the highest on the exam?
Algorithm	Quicksort
Program	Human-readable language (C++, Java…)
Instruction Set Architecture	Machine language
Microarchitecture	Hardware design
Logic	Set of rules
Digital Circuits	AND, OR, NOT …
Analog Circuits	Amplifiers, Filters …
Devices	Transistors (CMOS) ….
Physics	Electrons ….

[1] College Physics. Authored by: OpenStax College. Located at: https://openstax.org/books/college-physics/pages/1-introduction-to-science-and-the-realm-of-physics-physical-quantities-and-units. License: CC BY: Attribution. License terms: Located at License

What is a Computer System?



[1]



Problem

Algorithm

Program — Human-readable language (C++, Java…)

Instruction Set Architecture — Machine language

Microarchitecture — Hardware design

Logic — Set of rules

Digital Circuits — AND, OR, NOT …

Analog Circuits

Devices

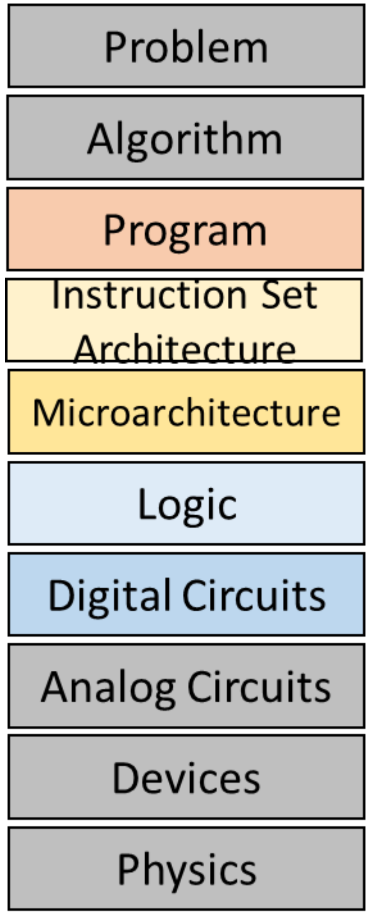
Physics

Focus of this course

[1] College Physics. Authored by: OpenStax College. Located at: https://openstax.org/books/college-physics/pages/1-introduction-to-science-and-the-realm-of-physics-physical-quantities-and-units. License: CC BY: Attribution. License terms: Located at License

Levels of Abstraction

- **Architecture (Instruction Set Architecture)**
  - A set of specifications that allows developers to write software and firmware
  - These include the **instruction set**.
- **Microarchitecture**
  - The logical organization of the inner structure of the computer
  - A specific implementation of the architecture
- **Hardware or Implementation**
  - The realization or the physical structure, i.e., **logic design** and chip packaging



Problem

Algorithm

Program

Instruction Set Architecture

Microarchitecture

Logic

Digital Circuits

Analog Circuits

Devices

Physics

Architecture Specifications

- The architecture specifies a contract between the hardware and software.
- Many different compatible processors may be implemented, e.g., to meet different power consumption, cost, area, and performance goals.
- When a software is written to conform with an architecture specification, it can be portable.

IBM System/360
Source of photo: Erik Pitti, CC-BY-2.0

Computer Architecture

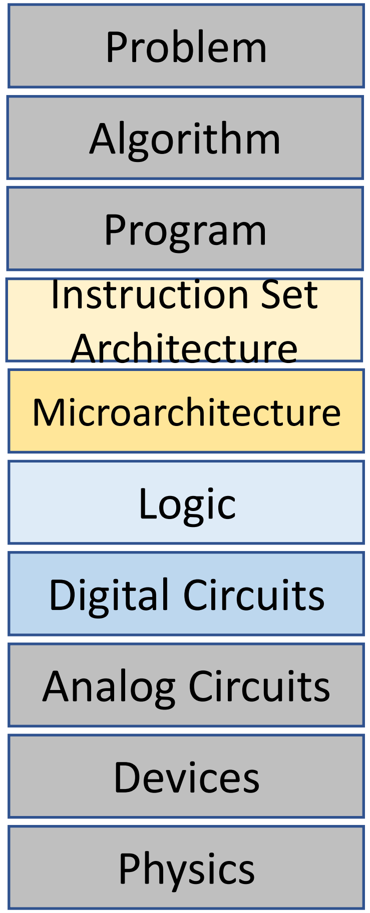
- Computer architecture is much more than the task of defining the instruction-set or high-level architecture.
- The computer architect must contribute to, and understand, all levels of design in order to deliver the most appropriate design for a particular application and target market.

Computer Architecture

- Computer architecture is concerned with how best to exploit fabrication technology to meet market demands.
    - *e.g., how best might we use five billion transistors and a power budget of two watts to design the chip at the heart of a mobile phone?*
- Computer architecture builds on a few simple concepts but is challenging as we must constantly seek new solutions.
- What constitutes the “best” design changes over time and depending on our use-case. It involves considering many different trade-offs.

Computer Architecture

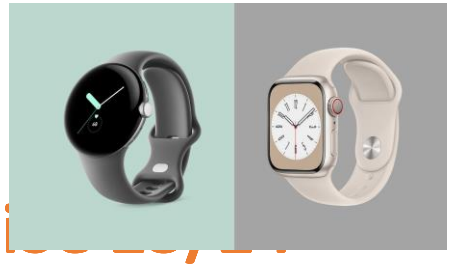
- Each level of design imposes different requirements and constraints, which change over time.
- History and economics: there is commercial pressure to evolve in a way that minimizes disruption and possible costs to the ecosystem (e.g., software).
- There is also a need to look forward and not design for yesterday’s technology and workloads!
- Design decisions should be carefully justified through experimentation.



Design Goals I

The market dictates the relative importance of different design goals – this often involves the computer architect having to make difficult trade-offs.

- **Functional** – hard to correct (unlike software). **Verification** is perhaps the highest single cost in the design process. We also need to Test our chips once they have been manufactured, again this can be a costly process and requires careful thought at the design stage.
- **Performance** – what does this mean? No single best answer, e.g., sports car vs. off-road 4x4 vehicle – performance will always depend on the “workload”, e.g., *Automotives, Mobile phones, smart watch etc*
- **Power** – a first-order design constraint for most designs today. Power limits the performance of most systems.

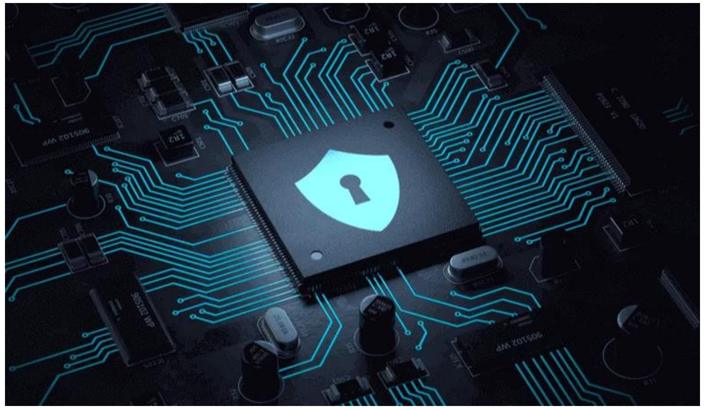


Universität Bremen

Design Goals II

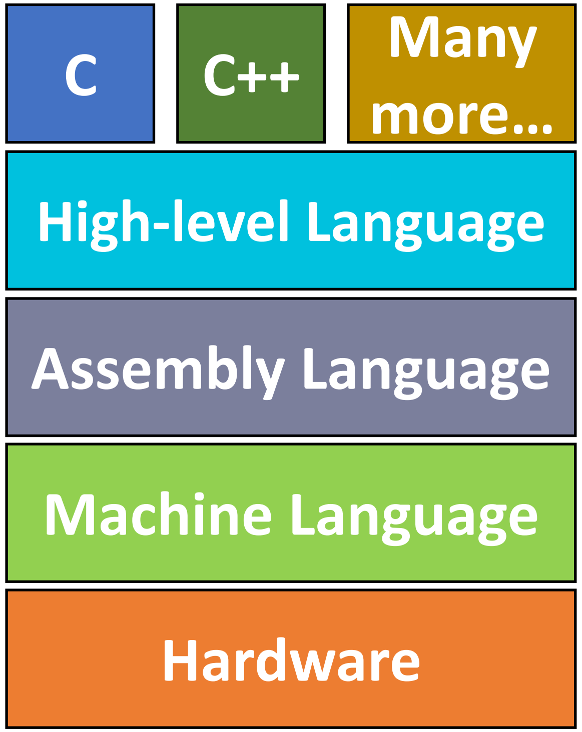
- **Security** – e.g., the ability to control access to sensitive data or prevent carefully crafted malicious inputs from hijacking control of the processor, e.g., ARM TrustZone.
- **Cost** – design cost (complexity), die costs (i.e., the size or area of our chip), packaging, etc.
- **Reliability** – do we need to try to detect and/or tolerate faults during operation?

***Pitfall:* Computer architecture becomes easy if we disregard some of our design constraints or goals**



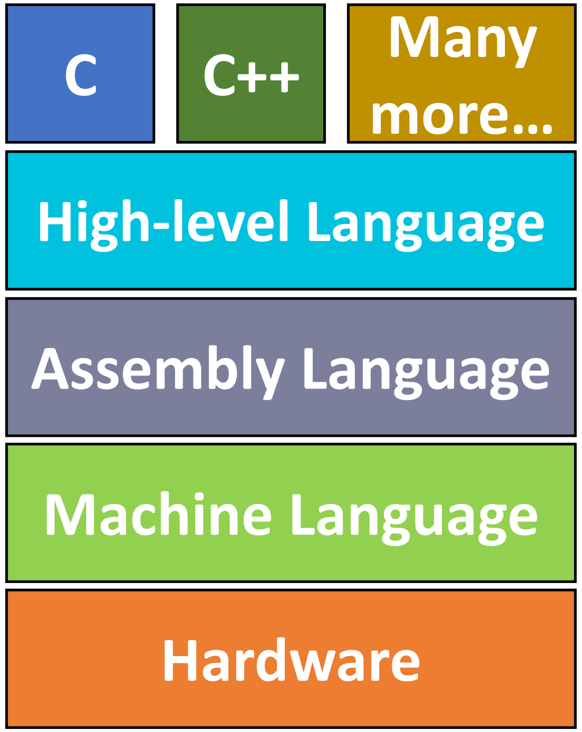
High-level Language to the HW Language

- The only language understood by computers is binary, also known as machine code
- Low-level languages
  - Assembly language
- High-level languages
  - C, C++, Rust, Python, many more …



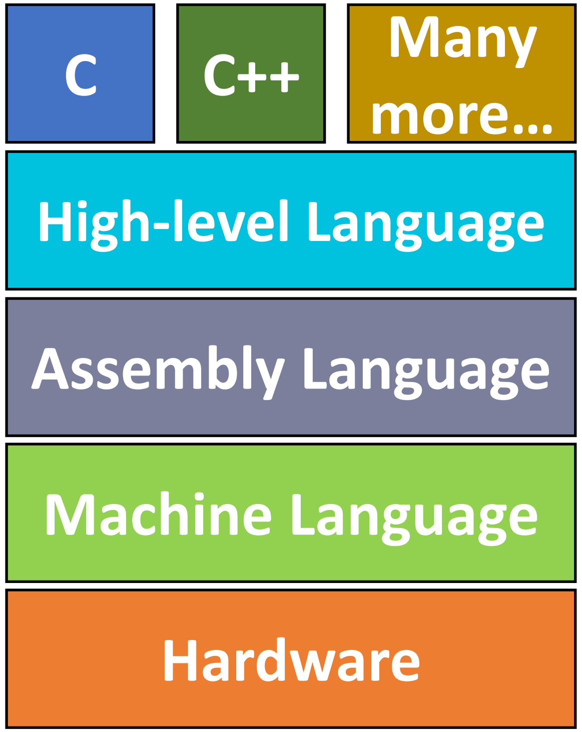
Low-level Language

- Low-level languages
  - To make it easier to program computers a programming language was invented
    - Assembly language
      - "MOV", "ADD" and "PUSH"
      - Convert to machine code – **Assembler**
  - More similar to machine code
  - Offer quicker ways to write binary
  - Hard to learn
  - Not very portable

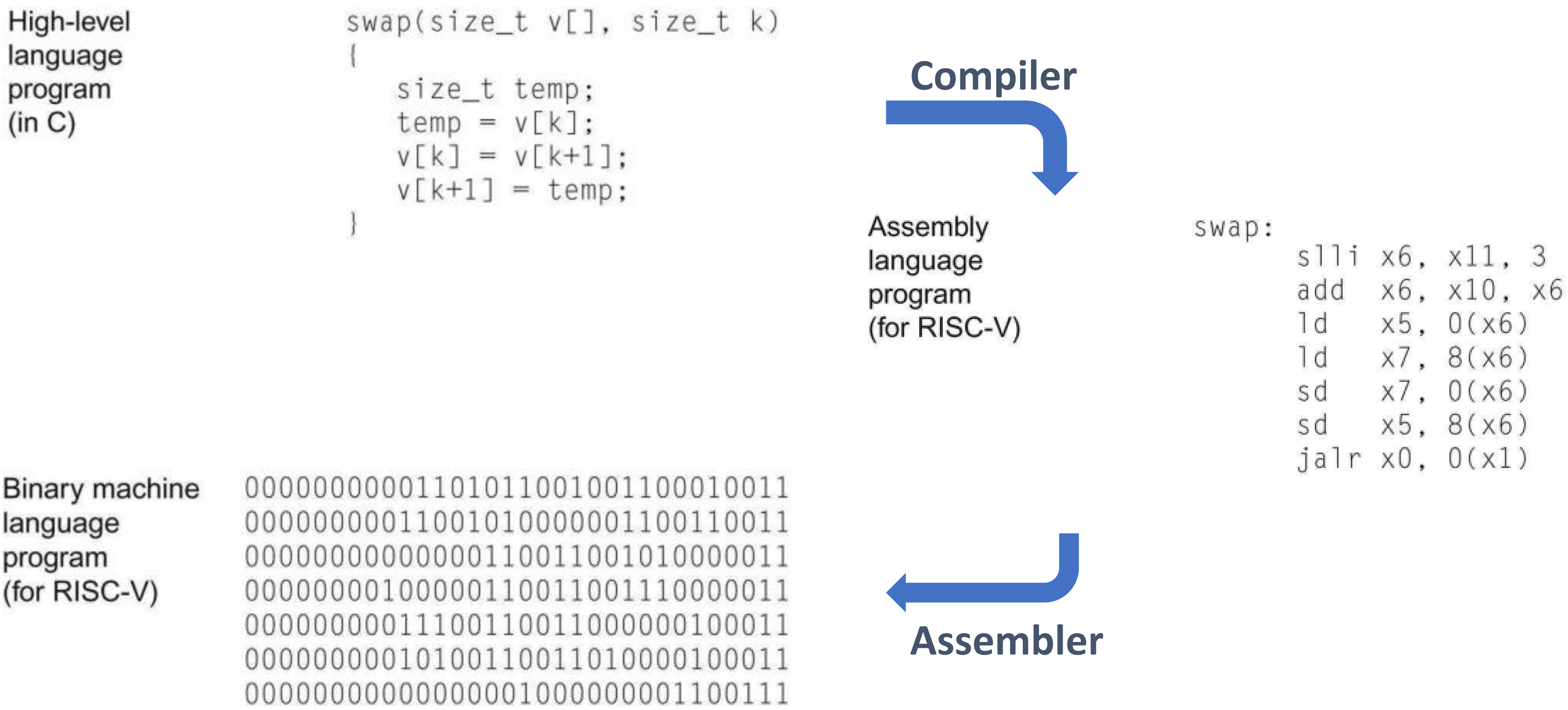


High-level Language

- High-level language
  - More developed than low-level languages
  - Examples: C, C++, Rust, Python, many more ….
  - Easier for humans to write, read, and maintain
  - Support for wide range of data types
  - Abstraction
    - They allow the programmer to think about how to solve the problem and not how to communicate with the computer.



High-level to Low-level Example



High-level language program (in C)

```
swap(size_t v[], size_t k)
{
    size_t temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compiler

Assembly language program (for RISC-V)

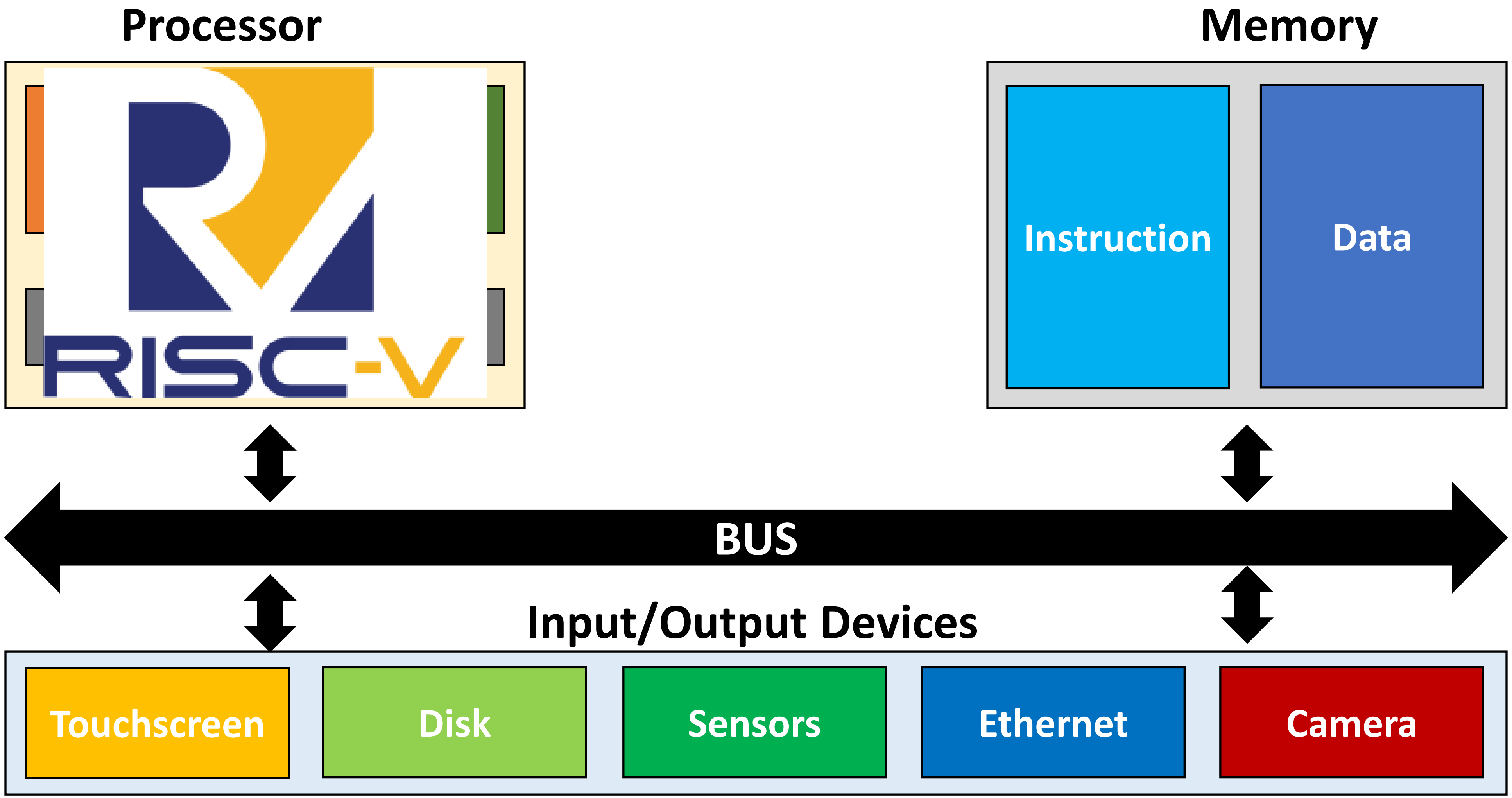
```
swap:
      slli x6, x11, 3
      add  x6, x10, x6
      ld   x5, 0(x6)
      ld   x7, 8(x6)
      sd   x7, 0(x6)
      sd   x5, 8(x6)
      jalr x0, 0(x1)
```

Assembler

Binary machine language program (for RISC-V)

```
00000000001101011001001100010011
00000000011001010000001100110011
00000000000000110011001010000011
00000000100000110011001110000011
00000000011100110011000000100011
00000000010100110011010000100011
00000000000000001000000001100111
```

Main Components of a Computer



Vectorised by Vulphere from https://riscv.org/wp-content/uploads/2017/05/Tue1100-RISC-V-Foundation-Update.pdf

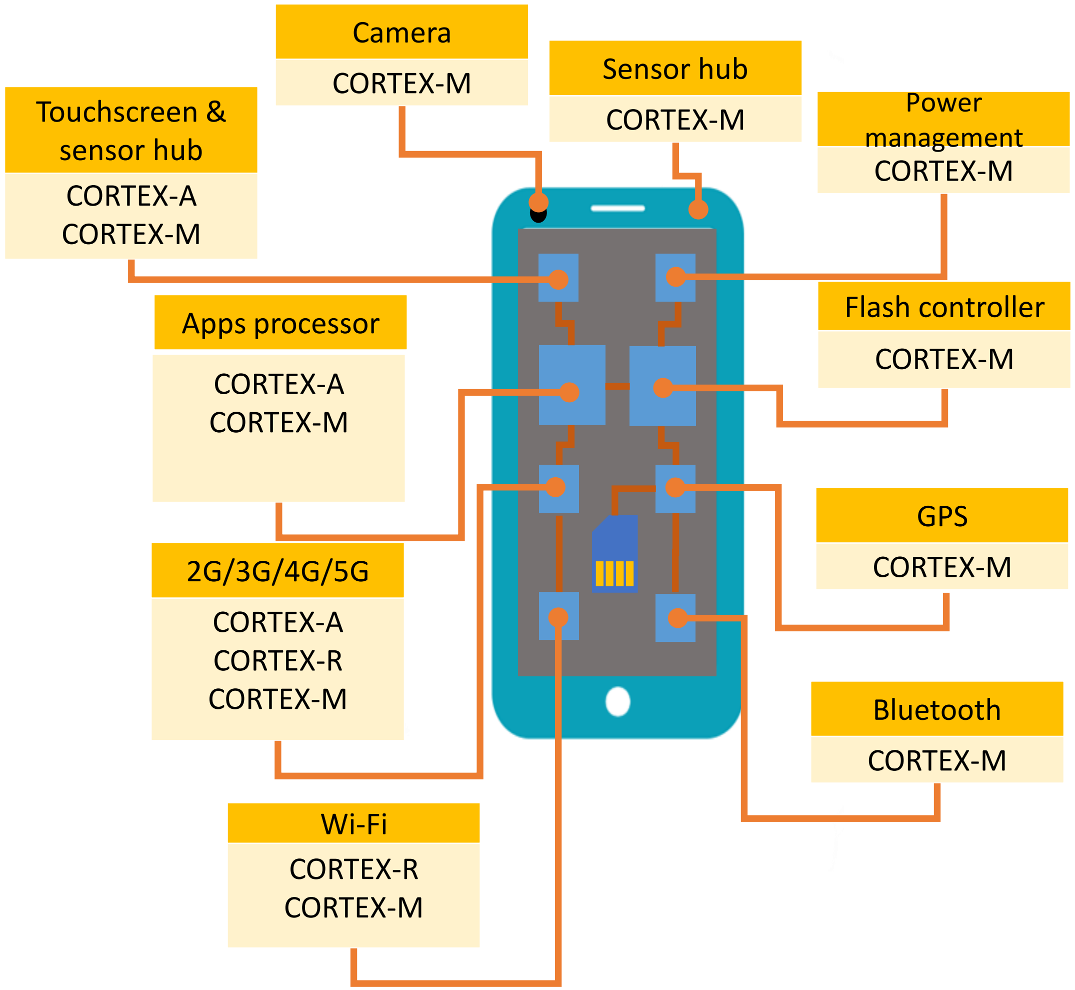
RISC-V

- Started as a “3-month project” in 2010 at UC Berkeley
  - Required a simple ISA which could be extended
  - Commercial ISAs were too complex and presented IP legal issues
- What is RISC-V?
  - A high-quality, license-free, royalty-free RISC ISA
  - Standard maintained by the non-profit RISC-V Foundation
  - Suitable for all types of computing systems
    - From Microcontrollers to Supercomputers
  - RISC-V is available freely under a permissive license
- RISC-V is not…
  - A Company
  - A CPU implementation

The Smartphone

- A single smartphone will contain multiple processor cores.

- *Why not use a single processor?*
  - *There are limits to how fast we can make a single processor*
  - Numerous smaller cores may consume less power
  - *Switch off as much of the chip as possible when it isn't in use*



Moore’s Law

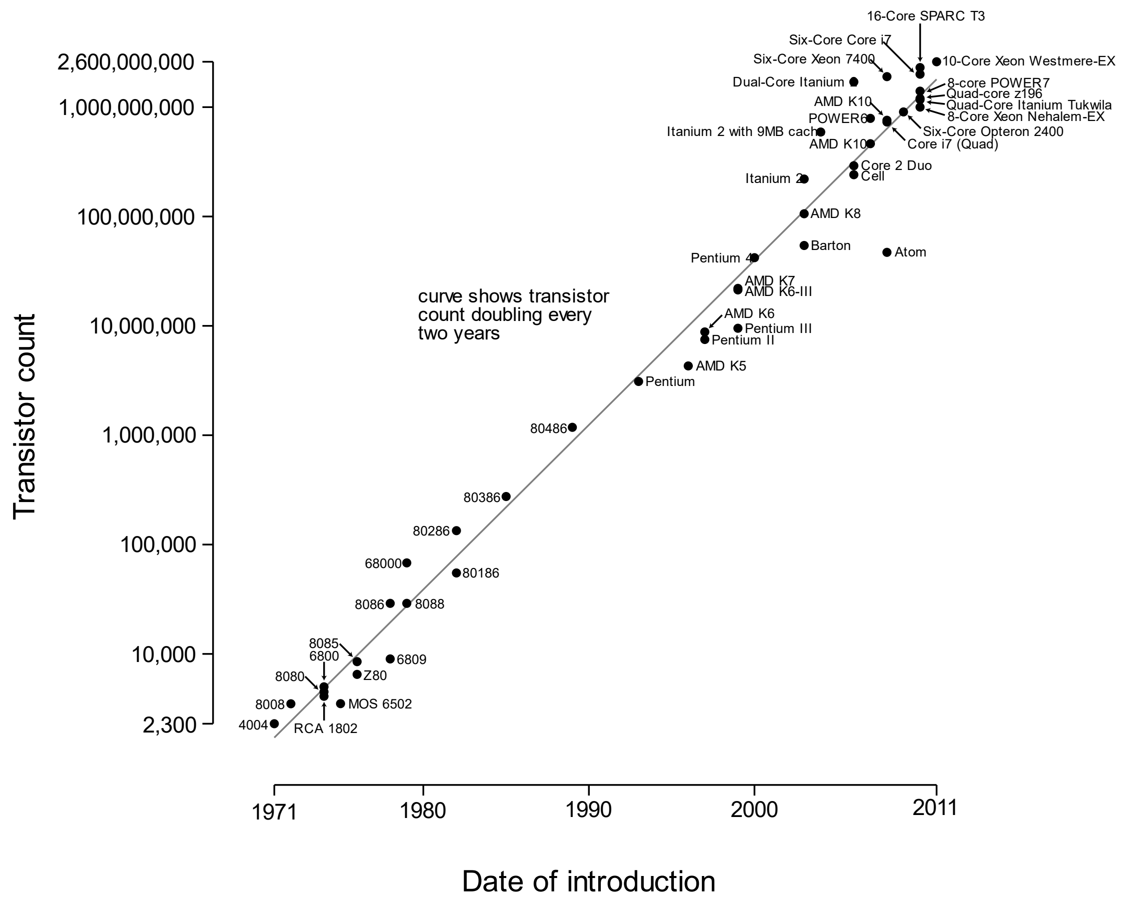
- In 1965, Intel co-founder Gordon Moore predicted the rapid increase in the number of transistors on a chip.
- By 1975, he had refined his prediction to what is now known as Moore’s Law
- Moore’s Law predicts that the number of transistors we can integrate onto a chip, for the same cost, doubles every 2 years.

Gordon Moore and Robert Noyce at Intel in 1970

Source: IntelFreePress, CC BY-SA-2.0

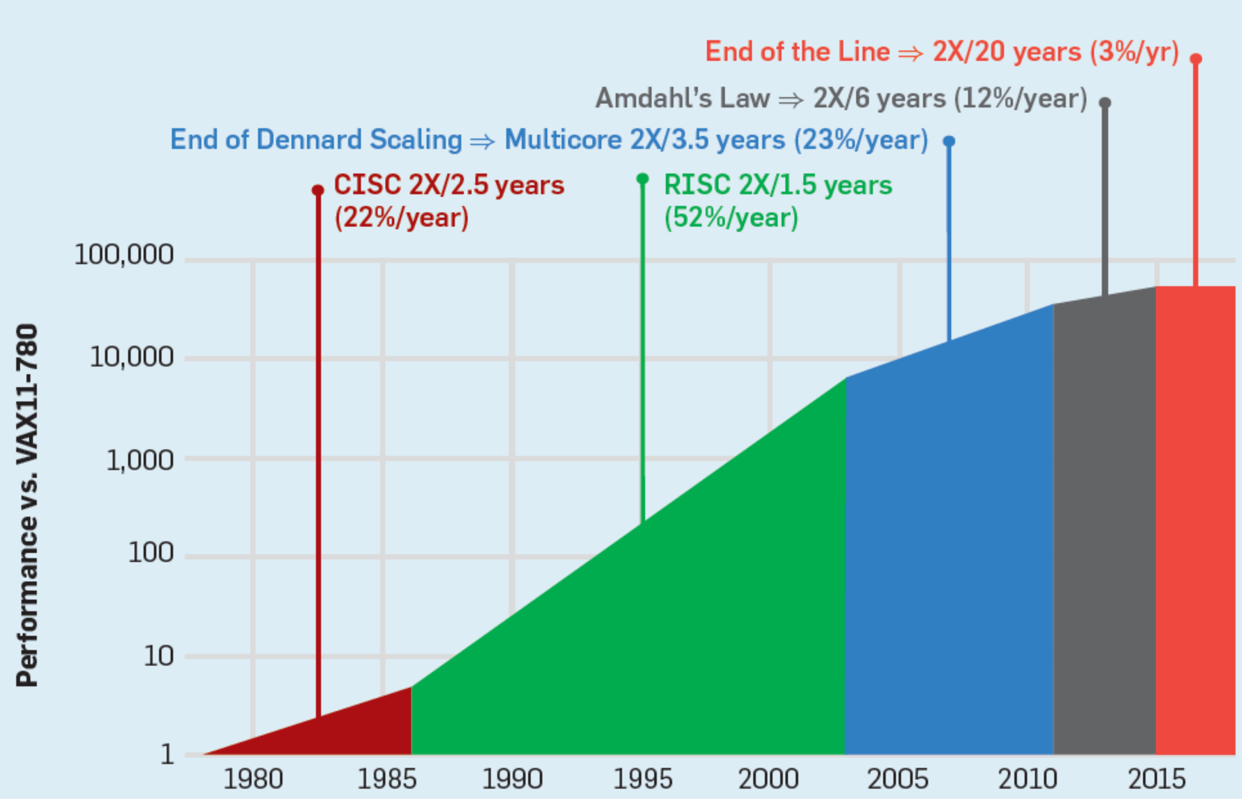
Moore’s Law

Microprocessor transistor counts 1971-2011 & Moore's law



Source: Wgsimon, Wikipedia, CC BY-SA 3.0

Increased performance didn’t come from pipelining and faster transistors alone



Source: Communications of the ACM 02/2019, „A New Golden Age for Computer Architecture“

Slowing Single-core Performance Gains

To summarize, sustaining single core performance gains became difficult due to:

- The limits of pipelining
- The limits of Instruction-Level Parallelism (ILP)
- Power consumption
- The performance of on-chip wires

No dependency – can be parallelized

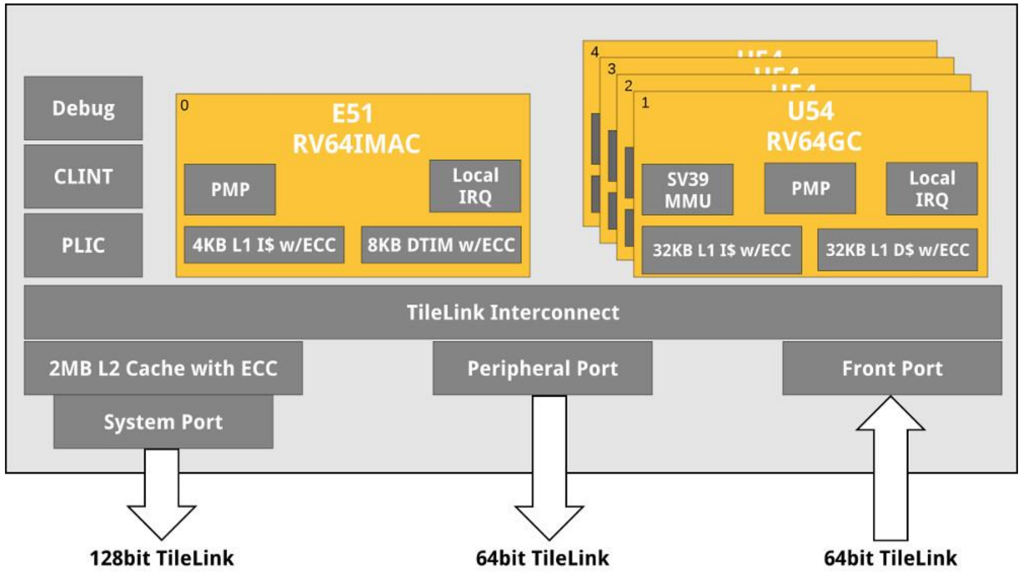
1. e = a + b
2. f = c + d
3. m = e * f

Depends on results of 1 and 2

As a result performance gains slowed from 52% to 21% per year for the highest performance processors.

Multicore Processors

- Eventually, it made sense to shift from single-core to multicore designs.
- From ~2005, multicore designs became mainstream.
- The number of cores on a single chip increased over time.
- Clock frequencies increased more slowly.
- Individual cores were designed to be as power efficient as possible.



e.g., 5 x RISC-V processors

Source: https://www.open-electronics.org/linux-ready-risc-v-64-bit-multicore-cpu-sifive-u54-mc-coreplex-ip/ (licensed under CC BY-SA-NC)

Multicore Processors

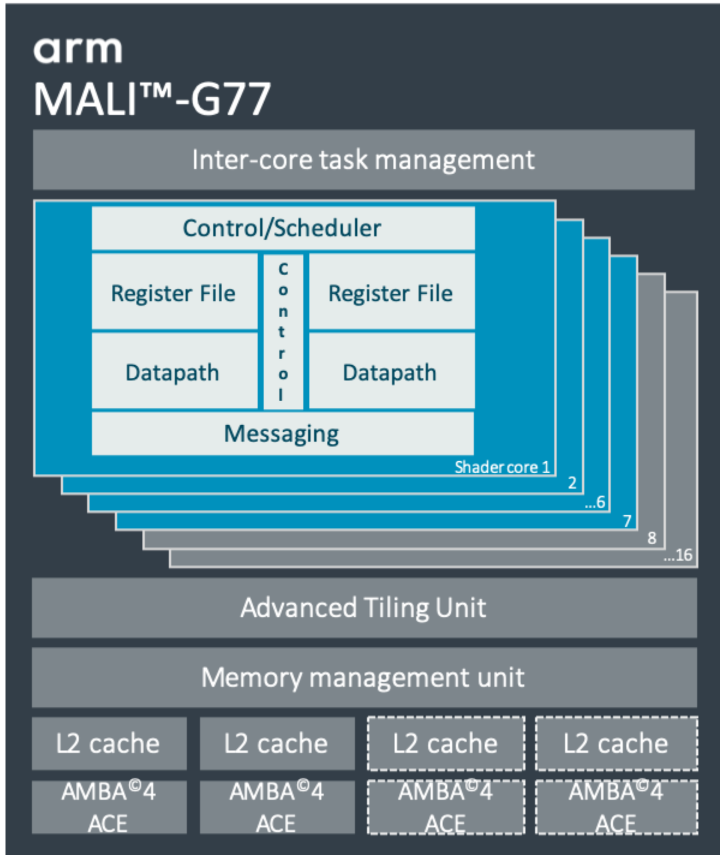
Exploiting multiple cores comes with its own set of challenges and limitations:

- Power consumption may still limit performance.
- We need to write scalable and correct parallel programs to exploit them.
- We might not be able to find enough parallel threads to take advantage of our cores.
- On-chip and off-chip communication will limit performance gains.
  - Off-chip bandwidth is limited and may throttle our many cores.
  - Cores also need to communicate to maintain a coherent view of memory.

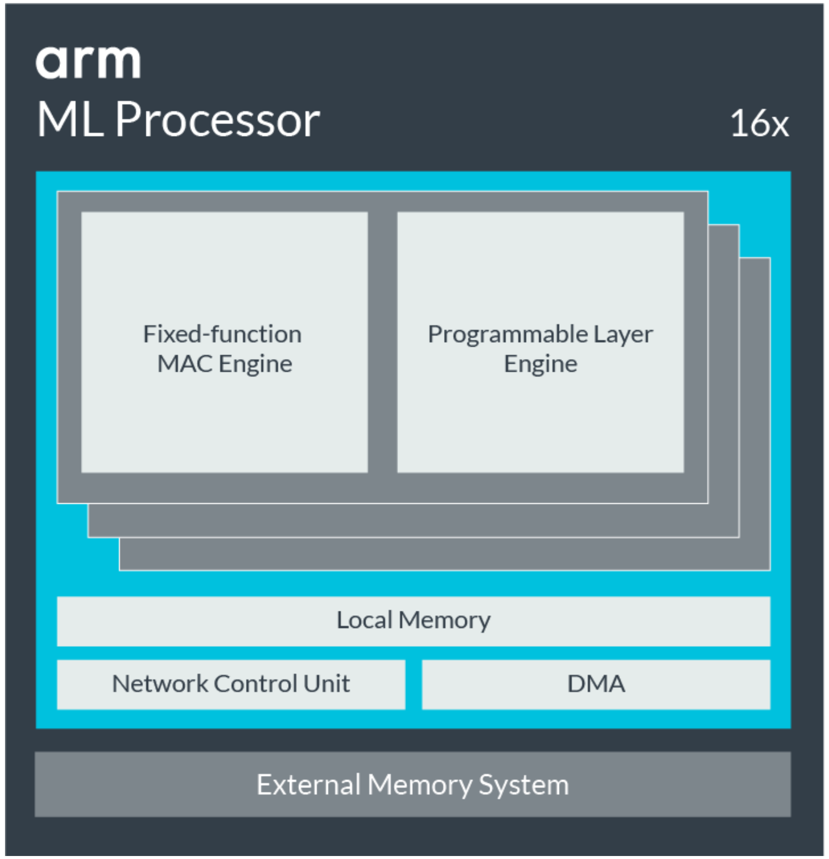
Specialization

- Today, we often need to look beyond general-purpose programmable processors to meet our design goals.
- We trade flexibility for efficiency.
- We remove the ability to run all programs and design for a narrow workload, perhaps even a single algorithm.
- These “accelerators” can be 10-1000x better than a general-purpose solution in terms of power and performance.

Specialization



Graphics Processing Unit (GPU)

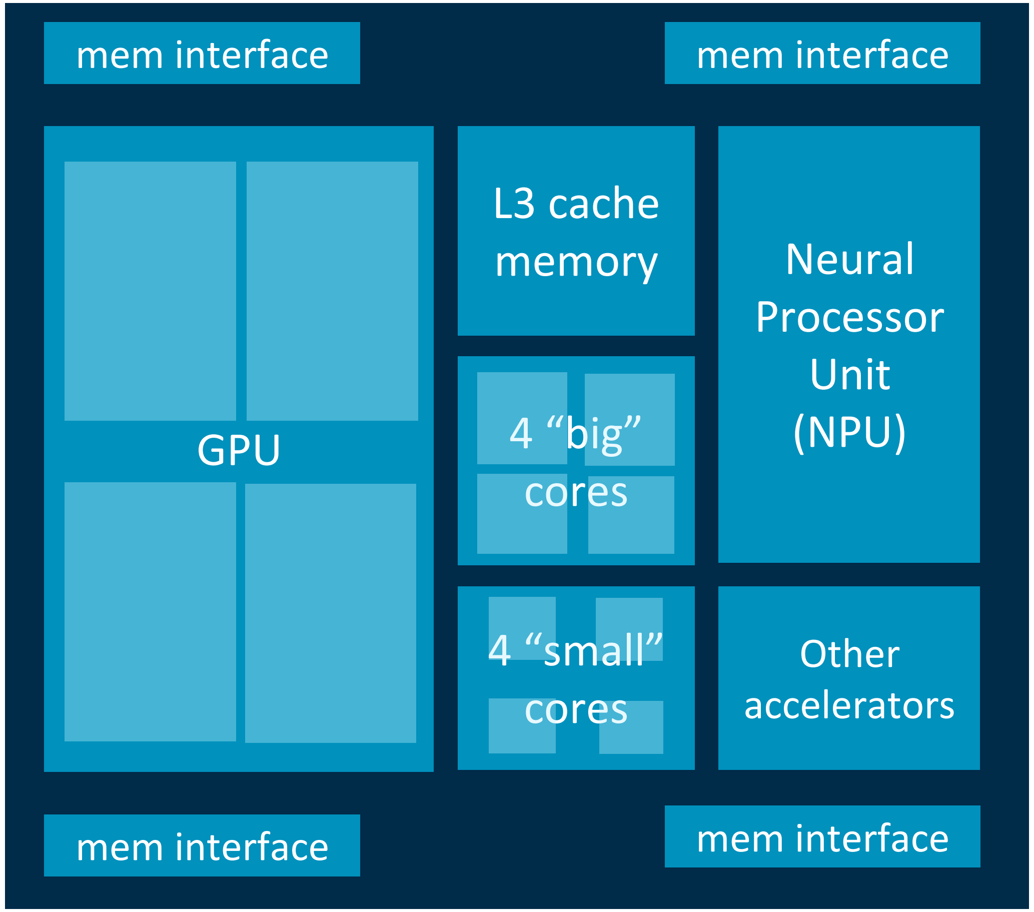


Neural Processor Unit (NPU)

Universität Bremen

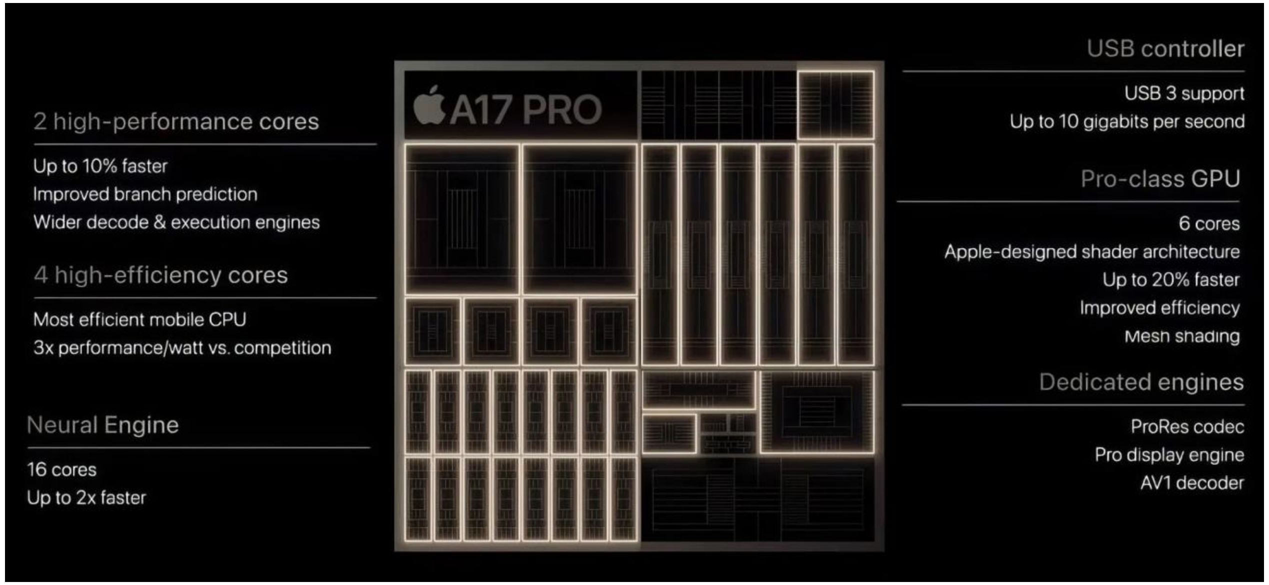
Today’s SoC Designs

- A modern mobile phone SoC (2019) may contain more than 7 billion transistors.
- It will integrate:
  - Multiple processor cores
  - A GPU
  - A large number of specialized accelerators
  - Large amounts of on-chip memory
  - High bandwidth interfaces to off-chip memory



A high-level block diagram of a mobile phone SoC

Modern SOCs Examples



2 high-performance cores

Up to 10% faster
Improved branch prediction
Wider decode & execution engines

4 high-efficiency cores

Most efficient mobile CPU
3x performance/watt vs. competition

Neural Engine

16 cores
Up to 2x faster

A17 PRO

USB controller

USB 3 support
Up to 10 gigabits per second

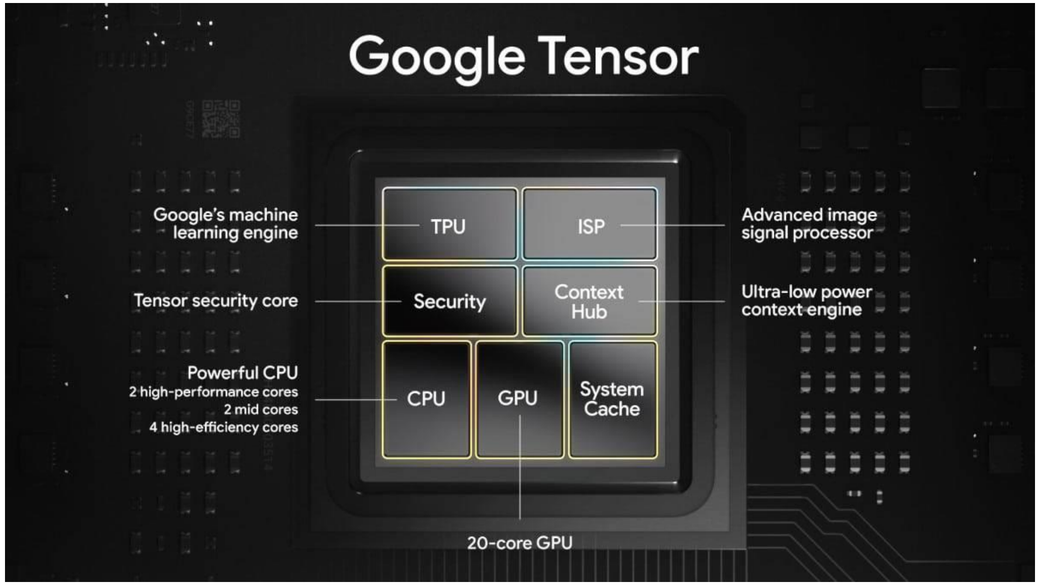
Pro-class GPU

6 cores
Apple-designed shader architecture
Up to 20% faster
Improved efficiency
Mesh shading

Dedicated engines

ProRes codec
Pro display engine
AV1 decoder

Apple A17 Pro



Google Tensor

Google's machine learning engine — TPU

ISP — Advanced image signal processor

Tensor security core — Security

Context Hub — Ultra-low power context engine

Powerful CPU
2 high-performance cores
2 mid cores
4 high-efficiency cores
— CPU

GPU — 20-core GPU

System Cache

Google Tensor

Trends in Computer Architecture

Time

Early computers	Gains from bit-level parallelism
Pipelining and superscalar issue	+ Instruction-level parallelism
Multicore/GPUs	+ Thread-level parallelism/data-level parallelism
Greater integration (large SoCs), heterogeneity, and specialization	+ Accelerator-level parallelism

Note: Memory hierarchy developments have also been significant. The memory hierarchy typically consumes a large fraction of the transistor budget.

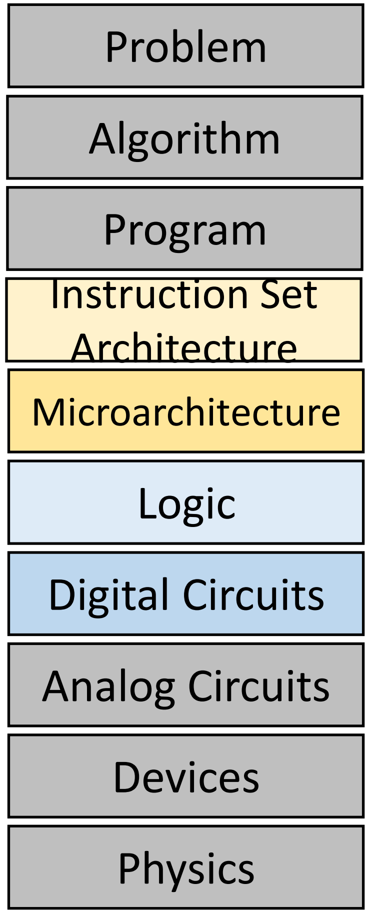
The Future – The End of Moore’s Law?

- The end of Moore’s Law has been predicted many times.
- Scaling has perhaps slowed in recent years, but transistor density continues to improve.
- Eventually, 2D scaling will have to slow down.
    - We are ultimately limited by the size of atoms!
- Where next?
    - Going 3D - Future designs may take advantage of multiple layers of transistors on a single chip.
        - Note: the gains are linear rather than exponential.
    - Better packaging and integration technologies (e.g., chip stacking)
    - New types of memory
    - New materials and devices

Universität
Bremen

An Introduction to Computer Architecture

Prof. Dr. Rolf Drechsler
Dr. Muhammad Hassan
M.Sc. Jan Zielasko
M.Sc. Milan Funck



Literature

- D. Patterson, J. Hennessy: Computer Organization and Design RISC-V Edition – The Hardware Software Interface, Elsevier, 2020.
- S. L. Harris, D. Harris: Digital Design and Computer Architecture, RISC-V Edition, Elsevier, 2021.
- ARM University program: Introduction-to-Computer-Architecture-Education.
- J. Teich, C. Haubelt: Digitale Hardware/Software-Systeme – Synthese und Optimierung, Springer Verlag, 2. Auflage, 2007.
- G. De Micheli: Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.
- G. D. Hachtel, F. Somenzi: Logic Synthesis and Verification Algorithms, Kluwer, 1996.

Literature

- H. Bähring, J. Dunkel, G. Rademacher: Mikrorechner-Systeme: Mikroprozessoren, Speicher, Peripherie, Springer-Verlag, 2. Auflage, 1994.
- J. P. Hayes: Computer Architecture and Organization, McGraw-Hill, 1998.
- M. G. Arnold: Verilog Digital Computer Design: Algorithms to Hardware, Prentice Hall, 1998.
- A. Tanenbaum: Structured Computer Organization, Prentice Hall, 5th Edition, 2006.
- D. A. Patterson, J. L. Hennessy: Computer Organization and Design - The Hardware/Software-Interface, Morgan Kaufmann, 3. Auflage, 2007.
- J. L. Hennessy, D. A. Patterson: Computer Architecture - A Quantitative Approach, Morgan Kaufmann, 4. Auflage, 2007.
- H. Kaeslin: Digital Integrated Circuit Design, From VLSI to CMOS Fabrication, Cambridge University Press, 2008.
- A. Biere, D. Kroening, G. Weissenbacher, C. M. Wintersteiger: Digitaltechnik – Eine praxisnahe Einführung, Springer-Verlag, 2008.