Inclusive Language Commitment

- *Arm is committed to making the language we use inclusive, meaningful, and respectful. Our goal is to remove and replace non-inclusive language from our vocabulary to reflect our values and represent our global ecosystem.*
- *Arm is working actively with our partners, standards bodies, and the wider ecosystem to adopt a consistent approach to the use of inclusive language and to eradicate and replace offensive terms. We recognise that this will take time. This course may contain references to non-inclusive language; it will be updated with newer terms as those terms are agreed and ratified with the wider community. We recognise that some of you will be accustomed to using the previous terms and may not immediately recognise their replacements. Please refer to the following examples:*
  - *When introducing the AMBA AXI Protocols, we will use the term 'Manager' instead of 'Master' and 'Subordinate' instead of 'Slave'.*
  - *When introducing the architecture, we will use the term 'Requester' instead of 'Master' and 'Completer' instead of 'Slave'.*
- *Contact us at education@arm.com with questions or comments about this course. You can also report non-inclusive and offensive terminology usage in Arm content at terms@arm.com.*

arm

An Introduction to Computer Architecture

Module 1

This Course

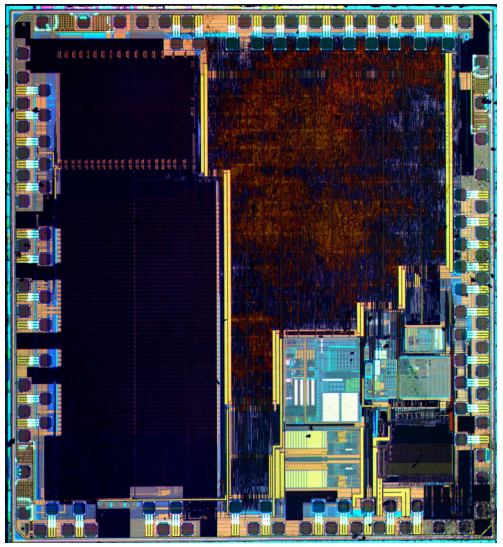
- Aims to introduce the key concepts and ideas in computer architecture
- Explores the design of modern microprocessors
- Examines important trends and current and future challenges

Module Syllabus

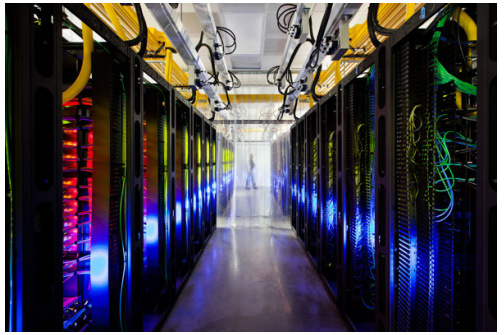
- What is computer architecture?
- Computer architecture arena and design goals
- Historical performance of computer architecture
- Future trends with multicore processors, systems on chip (SoCs), and beyond

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



Die photo of Arm Cortex-M3 Microcontroller with 16KB flash memory and 4KB RAM[1]

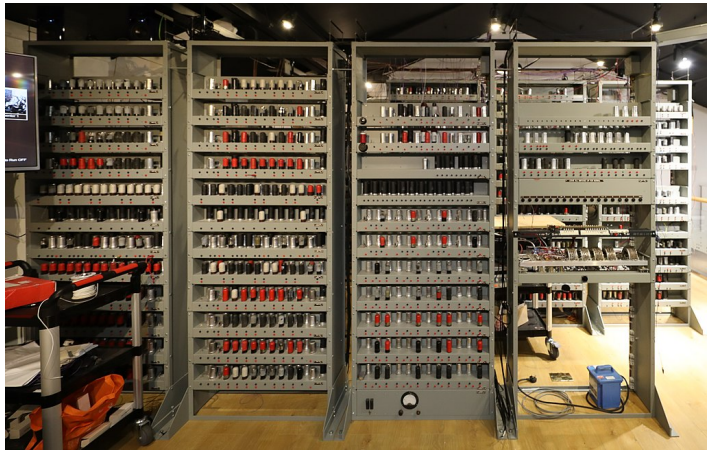


Google Data Center[2]

1. By Zeptobars, CC BY 3.0
2. By Connie Zhou, CC BY-NC 4.0

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]

BBC Micro Bit (2015) Arm Cortex-M0[2]

1. EDSAC photo, CC BY-SA 4.0
2. Aruld, CC BY-SA 4.0

Levels of Abstraction

- **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
- **Hardware or Implementation**
  - The realization or the physical structure, i.e., logic design and chip packaging

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 marketplace 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.

Markets

Applications

Operating Systems

Programming languages and compilers

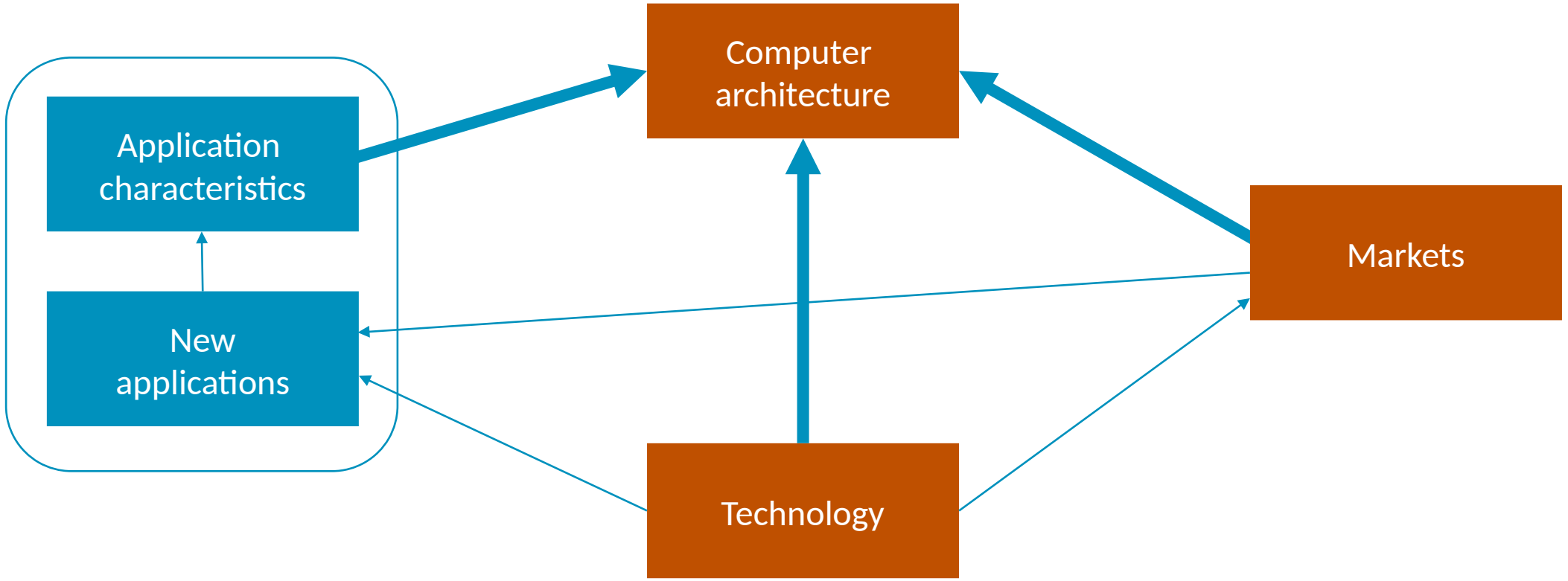
Architecture

Microarchitecture

Hardware

Fabrication Technology

The Computer Architecture Arena



Source: "Early 21st Century Processors," S. Vajapeyam and M. Valero, IEEE Computer, April 2004

arm

Design Goals I

- **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”
- **Power** – a first-order design constraint for most designs today. Power limits the performance of most systems.

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
- **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?

Markets and Features

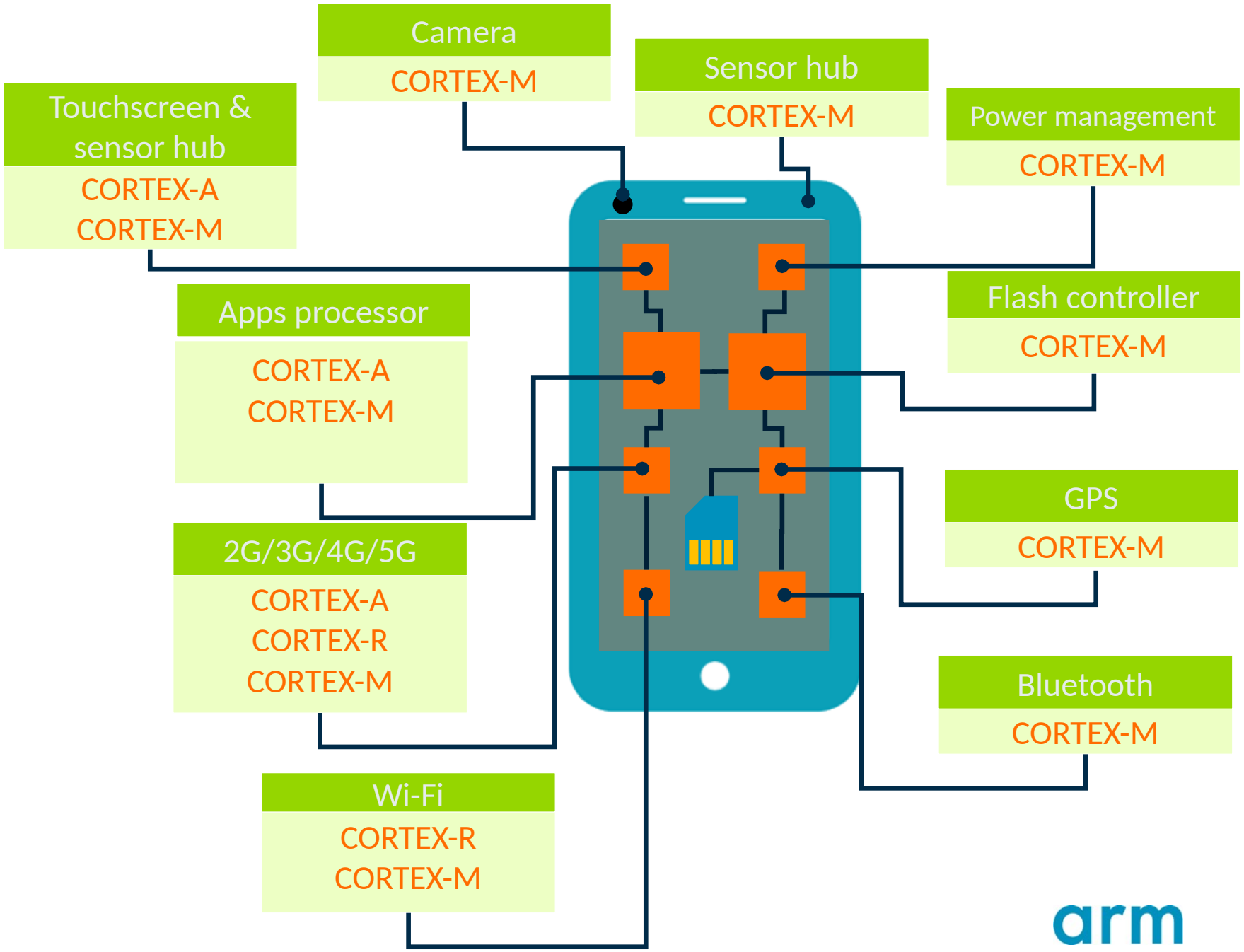
Each target market will require a different trade-off in terms of *power consumption*, *cost*, *area*, *performance*, *security*, *reliability*, etc.

Here are some example processor classes from Arm:

- **Cortex-A:** high-performance application processors, e.g., for mobile phones
- **Cortex-R**: deterministic real-time performance, fault detection, and tolerance.
- **Cortex-M**: energy-efficient embedded devices (“microcontroller” class cores)
- **Neoverse**: scalable networks of processors on a single chip
  - e.g., 8, 16, 64, or 128 cores. Used in datacenters, edge servers, and storage

The Smartphone

- A single smartphone will contain many different processor cores.
- *Why not use a single processor?*



Historical Performance

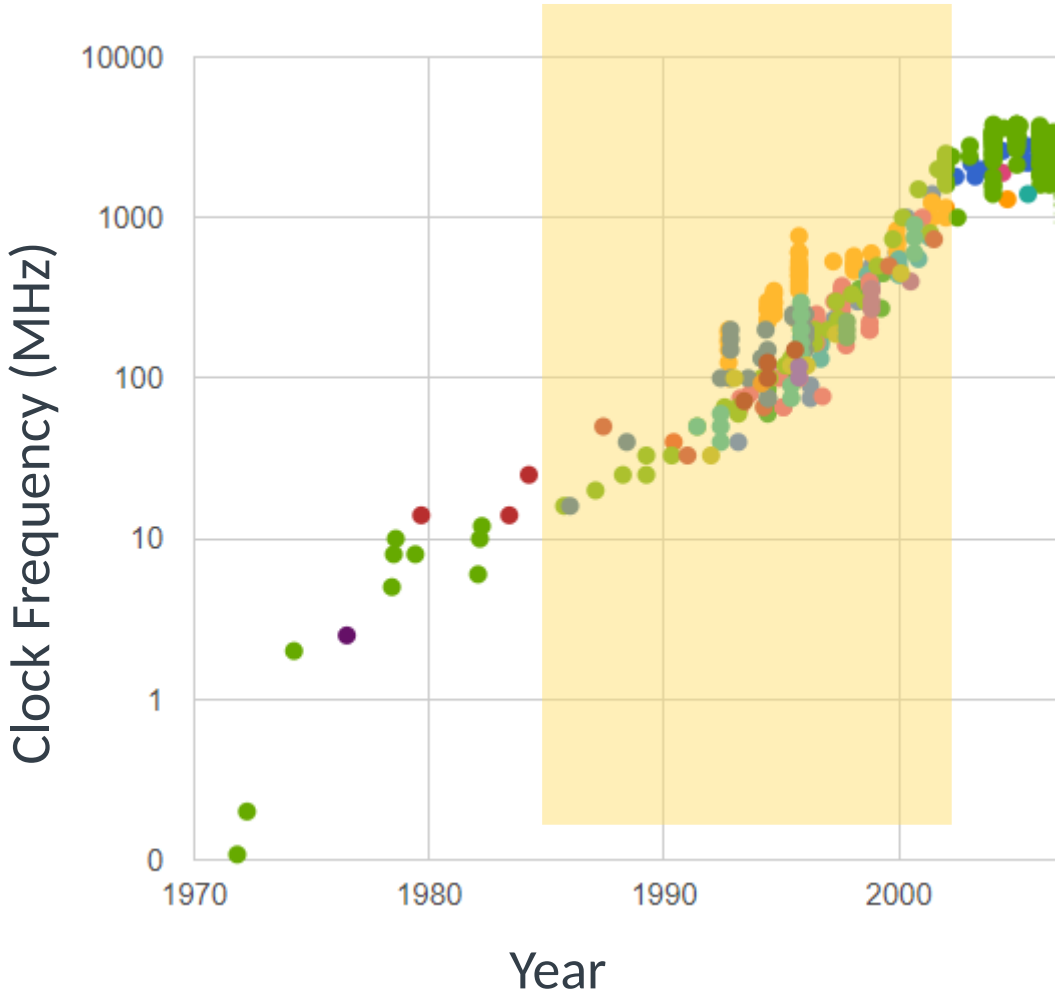
Historical Performance Gains

- By 1985, it was possible to integrate a complete microprocessor onto a single die or “chip.”
- As fabrication technology improved, and transistors got smaller, the performance of a single core improved quickly.
- Performance improved at the rate of 52% per year for nearly 20 years (measured using SPEC benchmark data).
- Note: the data are for desktop/server processors

Historical Performance Gains

Clock period

- Clock frequency improved quickly between 1985 and 2002:
  - ~10x from faster transistors, and
  - ~10x from pipelining and circuit-level advances.
- So overall, **~100X** of the total 800X gains came from reduced clock periods.



A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz. *Clock Frequency*, Stanford CPU DB. Accessed on Nov. 5, 2019. [Online]. Available: http://cpudb.stanford.edu/visualize/clock_frequency

Historical Performance Gains

- From **1985 to 2002,** performance improved by ~800 times.
- Over time, technology scaling provided much greater numbers of faster and lower power transistors.
- The “iron law” of processor performance:

Time = instructions executed x clocks per instruction (CPI) x clock period

- **Clocks per instruction (CPI)**
- We will also refer to **Instructions Per Cycle (IPC),** i.e., 1/CPI.

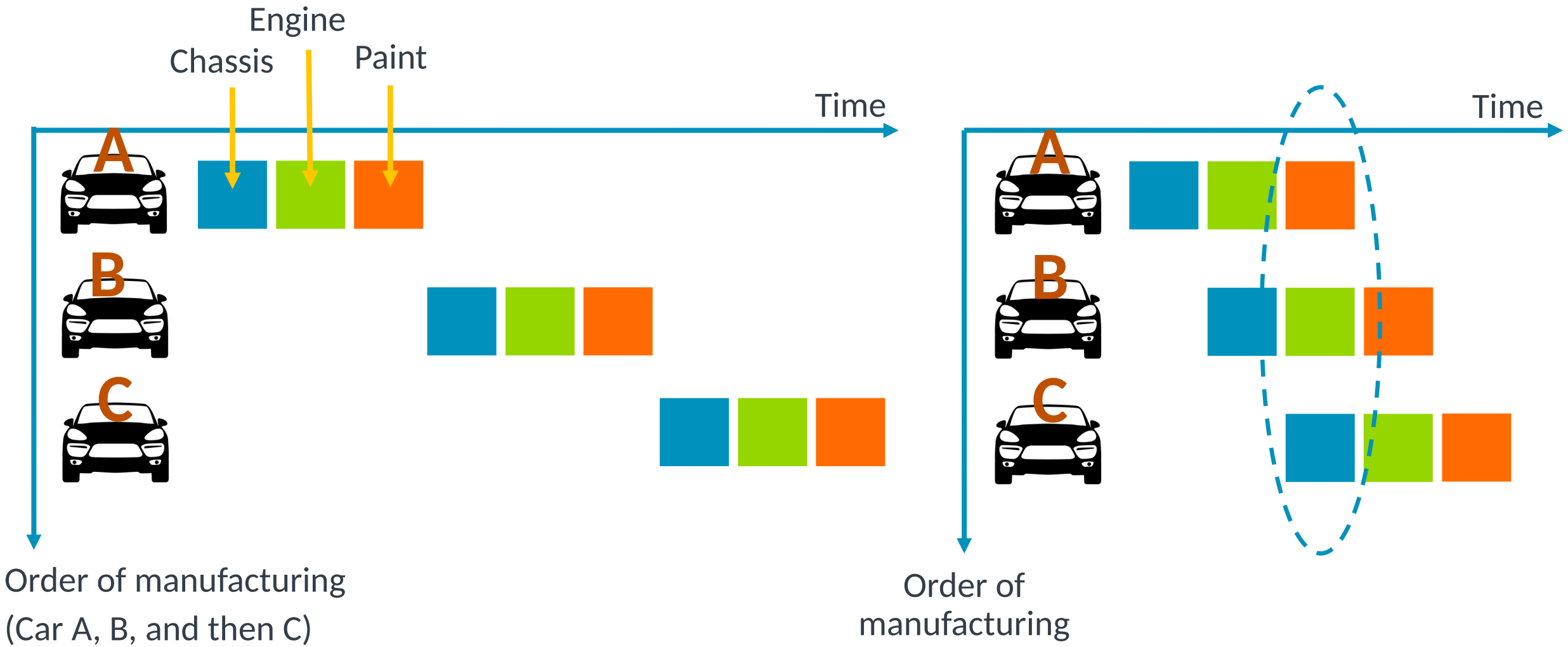
Clocks Per Instruction (CPI)

- Early machines were limited by transistor count. As a result, they often required multiple clock cycles to execute each instruction (CPI >> 1).
- As transistor budgets improved, we could aim to get closer to a CPI of 1.
- This is easy if we don’t care at all about clock frequency.
- Designing a high-frequency design with a good CPI is much harder. We need to keep our high-performance processor busy and avoid it stalling, which would increase our CPI. This requires many different techniques and costs transistors (area) and power.

Clocks Per Instruction (CPI)

- Eventually, the industry was also able to fetch and execute multiple instructions per clock cycle. This reduced CPI to below 1.
- When we fetch and execute multiple instructions together, we often refer to **Instructions Per Cycle (IPC),** which is 1/CPI.
- For instructions to be executed at the same time, they must be independent.
- Again, growing transistor budgets were exploited to help find and exploit this **Instruction-Level Parallelism (ILP).**

What Is Pipelining?

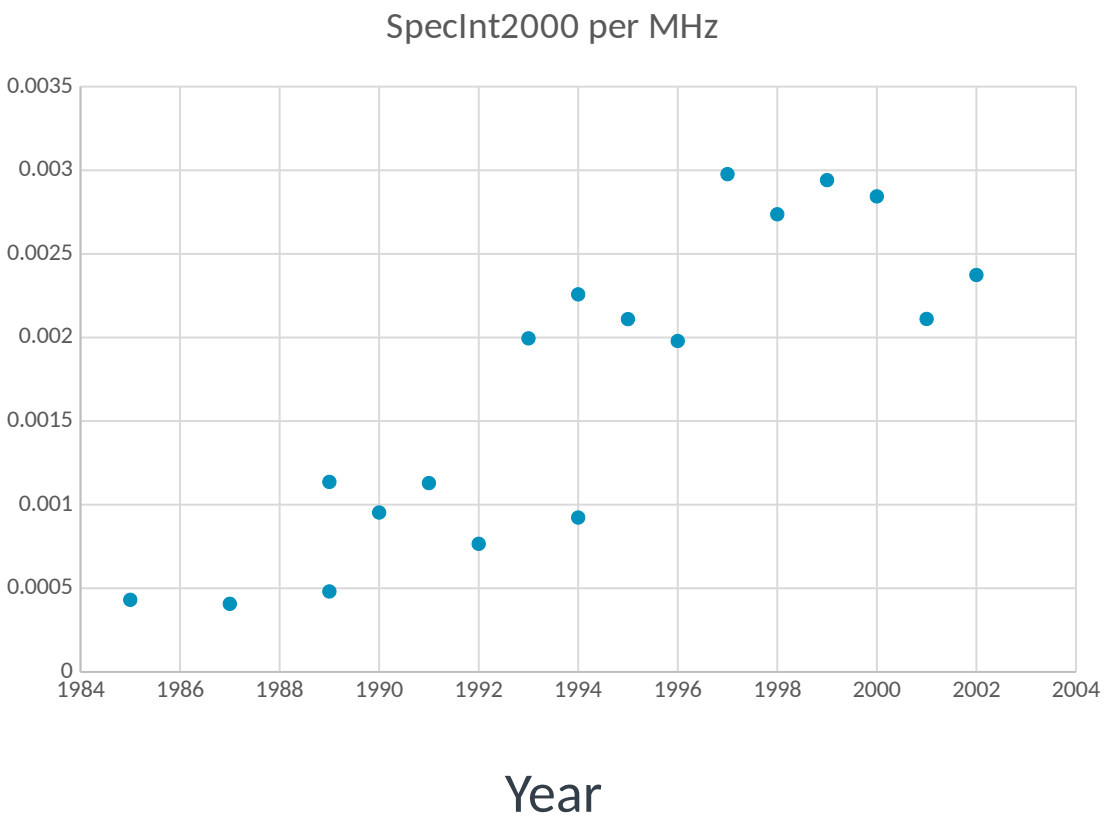


arm

IPC and Instruction Count

- Of the 800x improvement in performance (1985-2002), ~100x is from clock frequency improvements.
- The remaining gains (~8x) were from a reduction in instruction count, better compiler optimizations, and improvements in IPC.

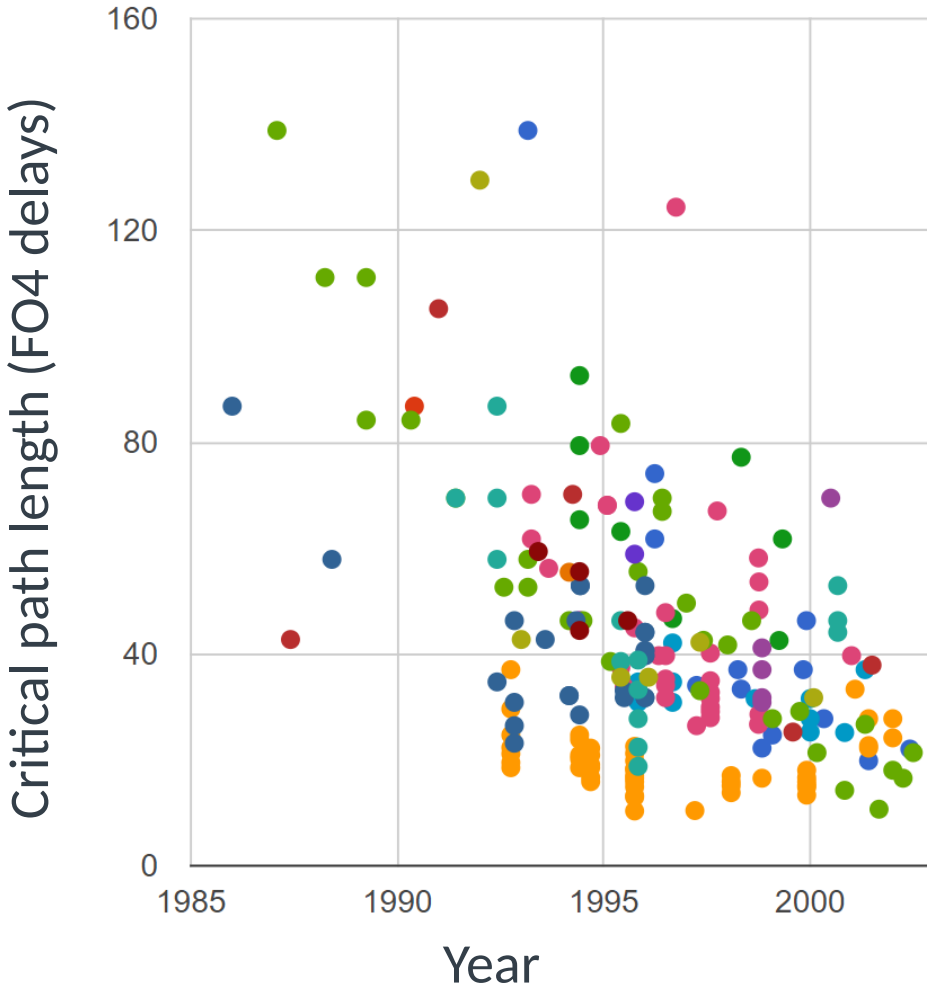
The graph to the right shows these improvements. It plots performance (SpecInt2000 benchmark performance per MHz for Intel processors against time).



arm

A Shorter Critical Path

- We can also try to reduce the number of gates on our critical path.
- This can be done by inserting additional registers to break complex logic into different “pipeline” stages.
- Advances were also made that improved circuit-level design techniques.
- The length of our critical paths reduced by **~10x** (1985-2002).



A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz. Stanford CPU DB. Accessed on Nov. 5, 2019. [Online]. Available: http://cpudb.stanford.edu

Moore’s Law

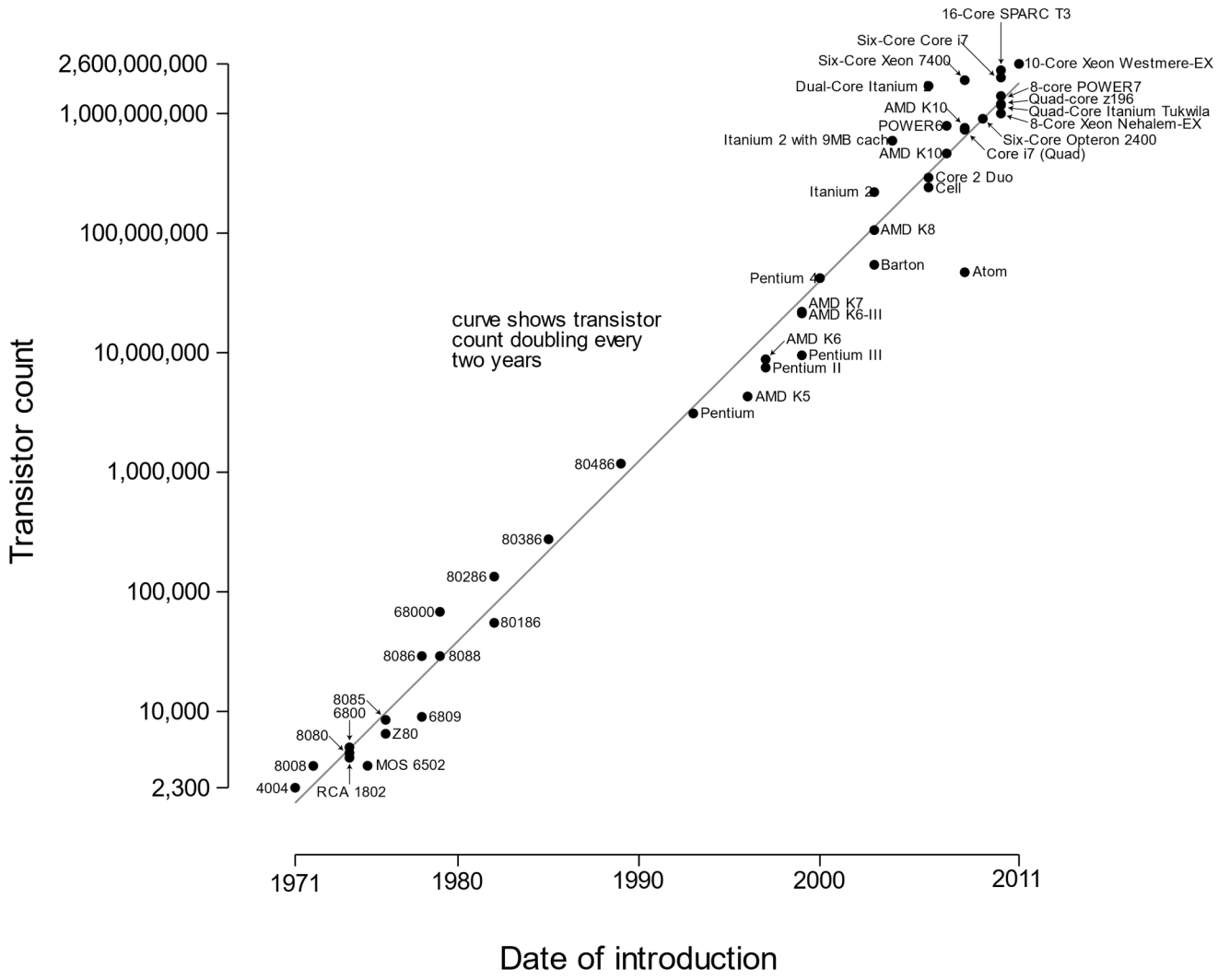
- Moore’s Law predicts that the number of transistors we can integrate onto a chip, for the same cost, doubles every 2 years.

Gordon Moore and Robert Noyce at Intel in 1970

Source: IntelFreePress, CC BY-SA-2.0

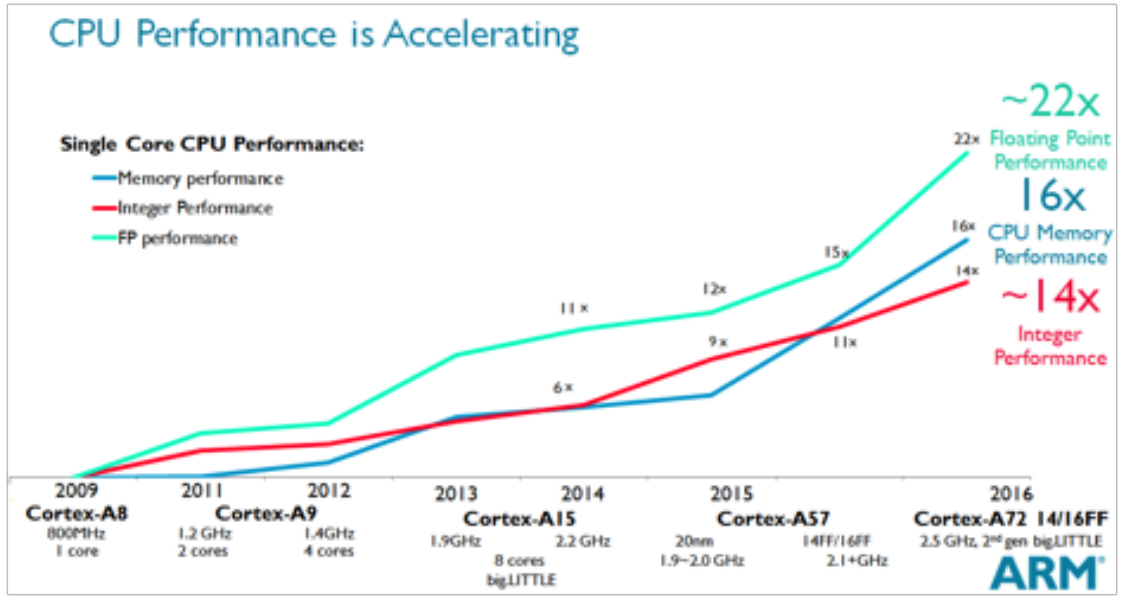
Moore's Law

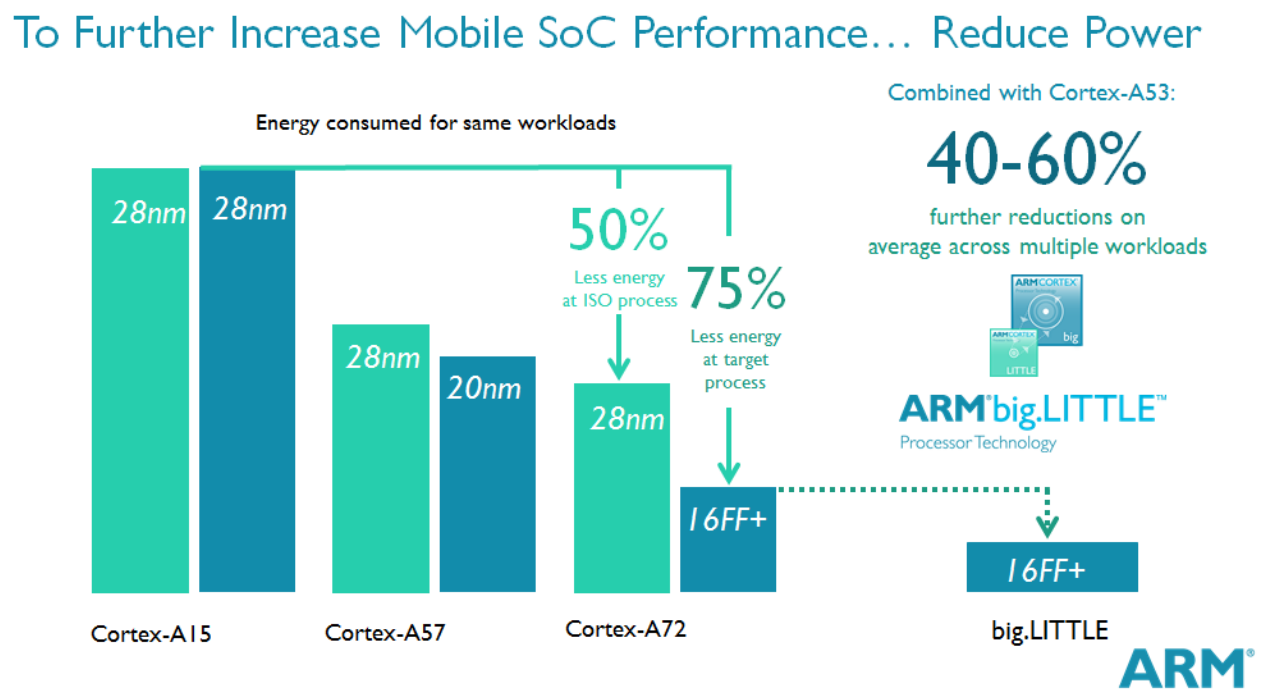
- Processor transistor budgets grew quickly as microarchitectures became more complex.
- **1985 – Intel 386**
  275K transistors, die size = 43 mm^2
- **2002 – Intel Pentium 4**
  42M transistors, die size = 217 mm^2



Source: Wgsimon, Wikipedia, CC BY-SA 3.0

Better Performance and Lower Power





arm

Multicore Processors, Systems on Chip (SoC), and Beyond

Slowing Single-core Performance Gains

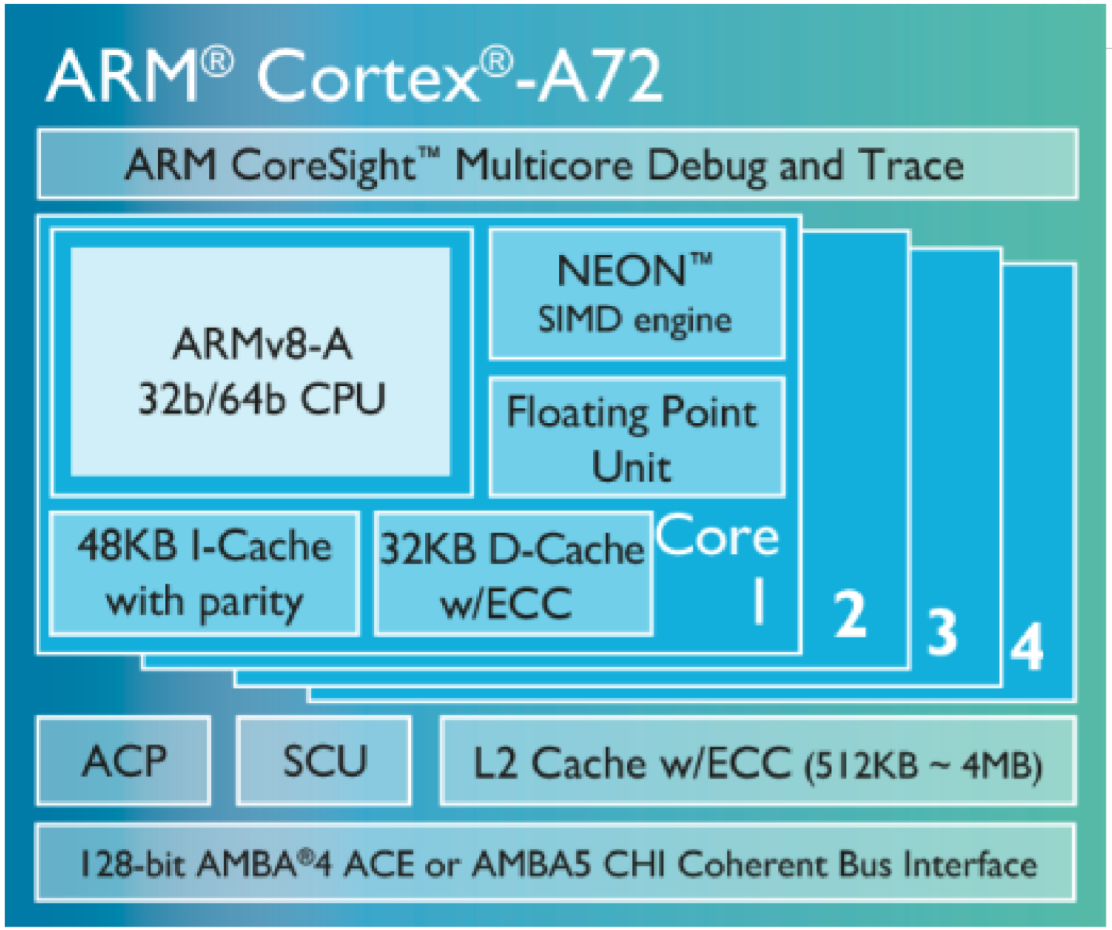
To summarize, sustaining single core performance gains became difficult due to:

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

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., 4 x Arm Cortex-A72 processors, each with their own L1 caches and a shared L2 cache

arm

Multicore Processors

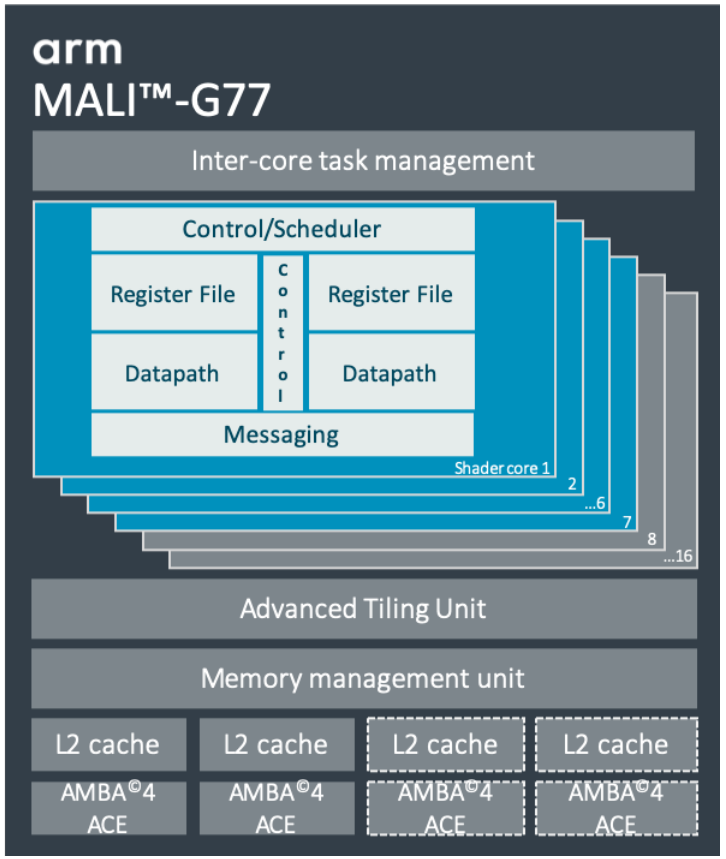
Exploiting multiple cores comes with its own set of challenges and limitations:

- 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.

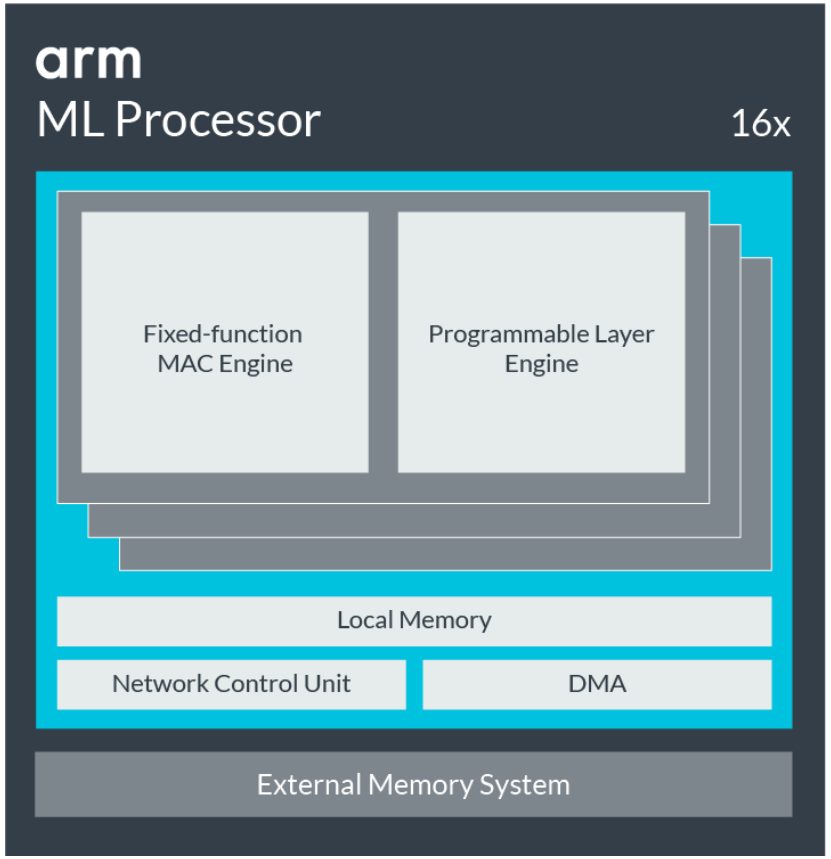
Specialization

- 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)

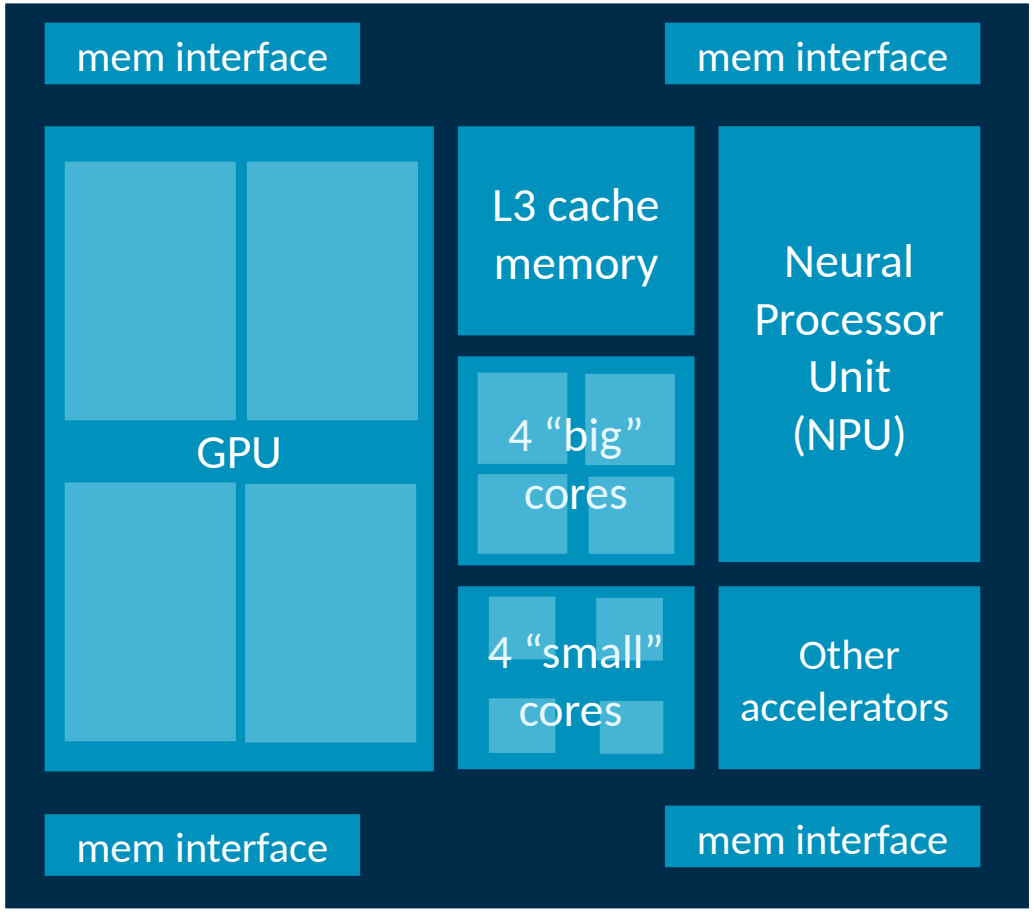


Neural Processor Unit (NPU)

arm

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

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

Backup Slides

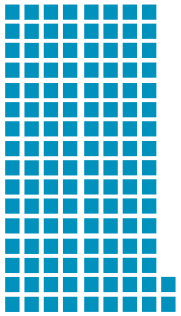
From Sensors and Smartphones to Servers

An area optimized microcontroller core (e.g., Arm Cortex-M0)

1X

1 square represents the **area** of this core

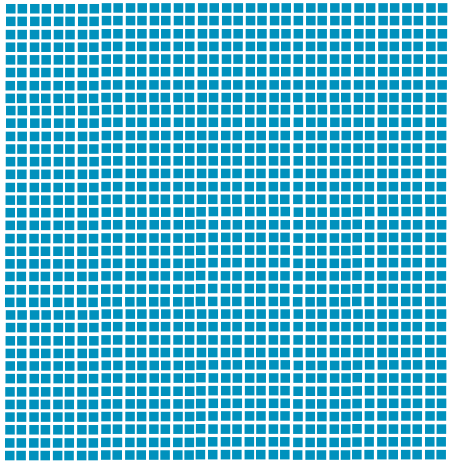
Mid-range 64-bit processor (e.g., Arm Cortex-A55). For smartphones, TVs, network infrastructure, …



130X

1 laptop or server class processor (e.g., A76 core with 512KB of L2 cache)

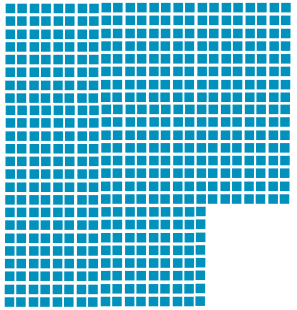
1380X



High-performance 32-bit core (e.g., Arm Cortex-M7)

Used in automotive, sensor hub, and other embedded applications.

13X



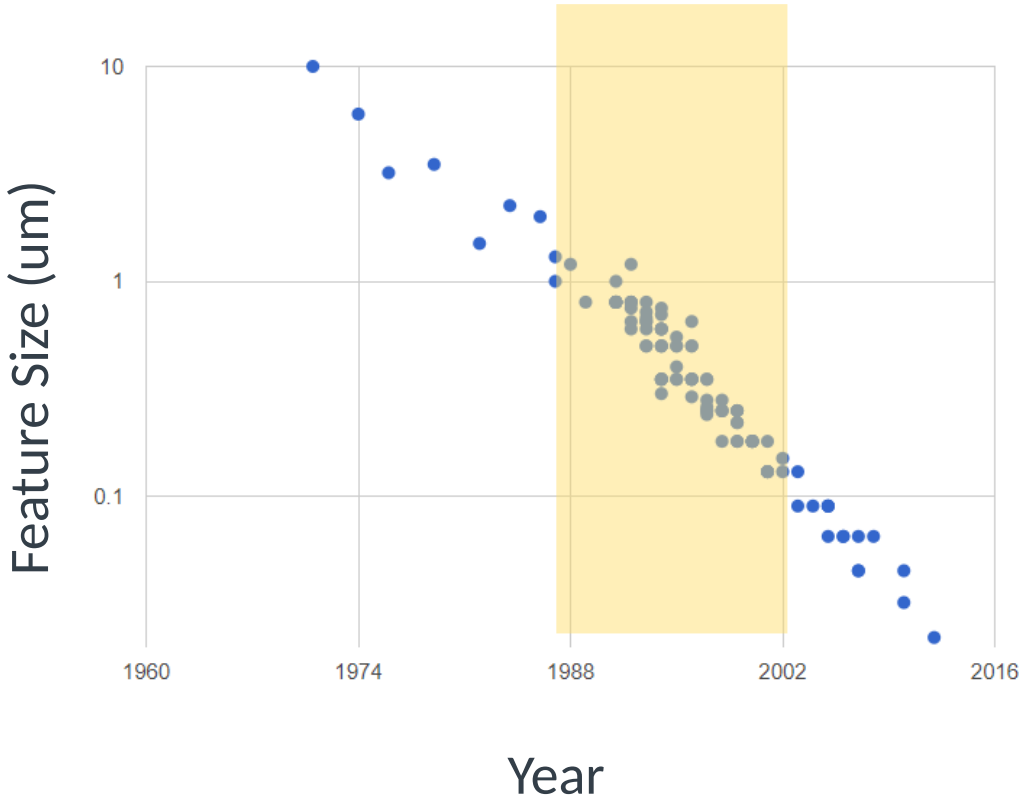
520X

High-performance processor (e.g., Arm Cortex-A73). For mobile and consumer devices.

arm

Technology Scaling: Faster Transistors

- From 1985 to 2002, we saw ~7 new process generations.
- Scaling provides smaller and faster transistors. Performance improves ~1.4x per generation, so for 7 generations, we have **~10x faster logic gates.**



A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz. Stanford CPU DB. Accessed on Nov. 5, 2019. [Online]. Available: http://cpudb.stanford.edu

arm

Instruction Count

- Increased datapath width (e.g., 16-bit to 32-bit to 64-bit)
- Larger register files (fewer load/store instructions)
- More complex instructions?
- SIMD instructions

Limits to Single Core Performance

- **Limits to pipelining**
  - Cost of interruptions grow, e.g., impact of cache misses and mispredicted branches.
  - Ultimately, some components are difficult or expensive to pipeline.
  - There are also practical limits to distributing very high-frequency clocks, registers represent a finite delay, and we may struggle to balance logic between pipeline stages.
- **Limits of Instruction-Level Parallelism (ILP)**
  - Large amounts of ILP are very difficult to discover and exploit efficiently.
  - Our returns on investment quickly diminish, i.e., we must use more power and more transistors to expose and exploit ever smaller amounts of ILP.

Limits to Single Core Performance

- **Power consumption**
  - Historical performance gains have been impressive, but power consumption also grew very quickly during the 1980s and 1990s.
  - This happened even with improvements in fabrication technology and reductions in supply voltage.
  - Power quickly became, and remains, a first-order design constraint for all significant markets.

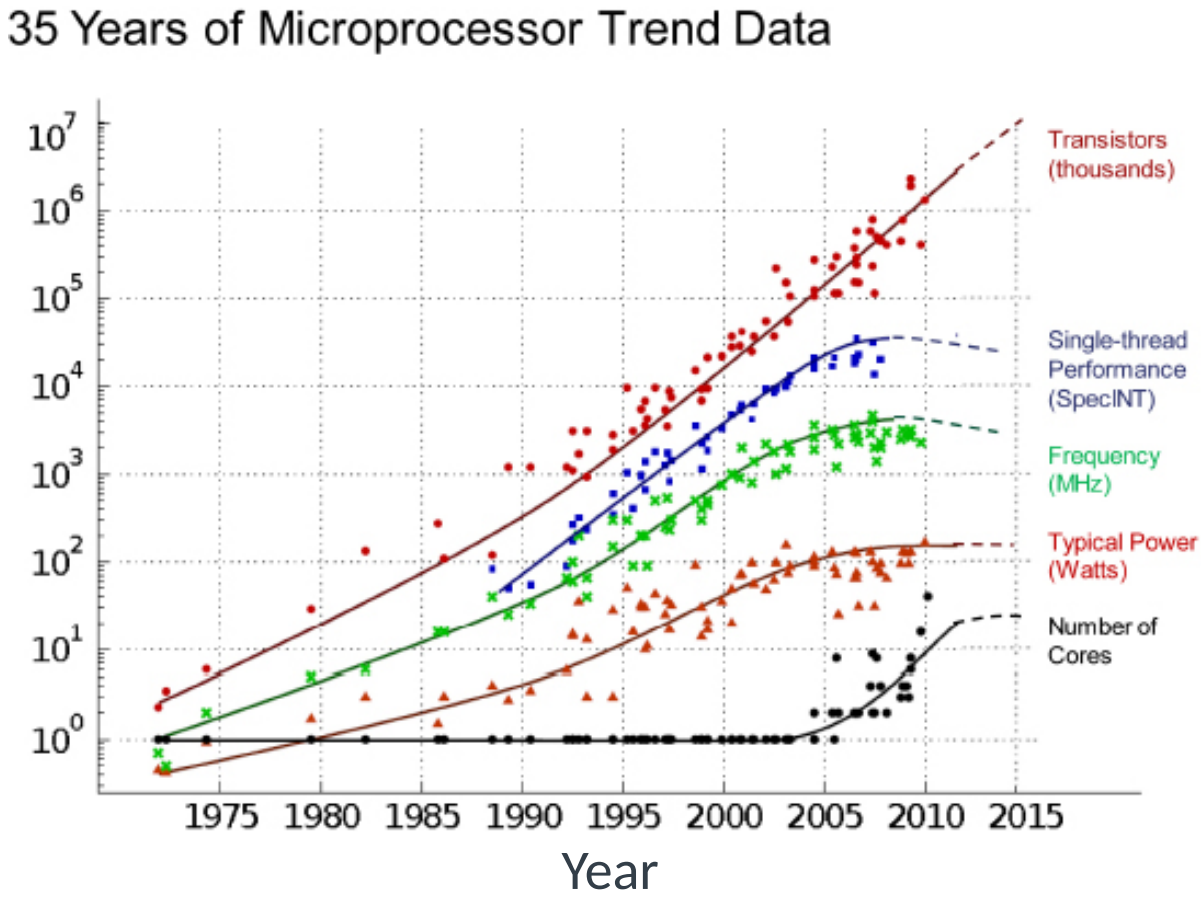


Figure source: Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. Dotted-line extrapolations by C. Moore: Chuck Moore, 2011, "Data processing in exascale-class computer systems," *The Salishan Conference on High Speed Computing*, April 27, 2011.

arm

Limits to Single Core Performance

- **On-chip wiring**
  - Wire delays scale relatively poorly compared to logic delays.
  - This limits the amount of state reachable in one clock cycle.
  - Unfortunately, this limits the performance of large complex processors.

Specialization

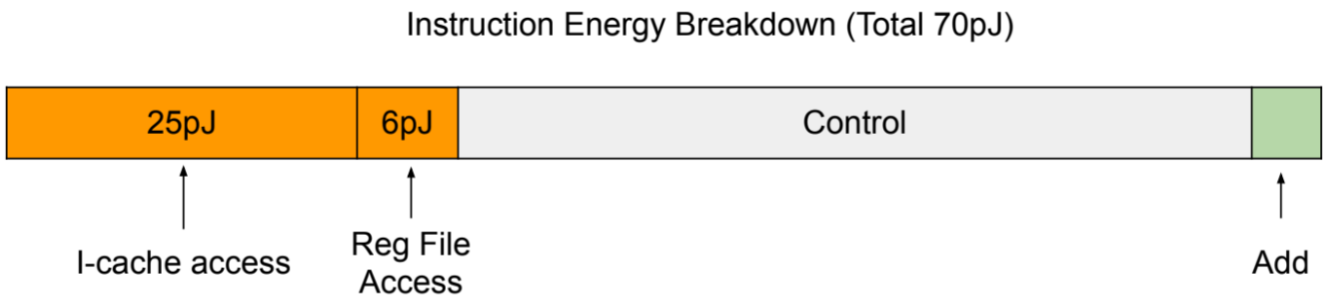
What does specialization allow us to do?

- Remove infrequently used parts of the processor.
- Tune the instruction set for common operations or replace with hardwired control .
- Exploit forms of parallelism abundant in the application(s) – we often see a specialized processing element and local memory reproduced many times.
- Instantiate specialized memories and tune their widths and sizes.
- Provide specialized interconnect between components.
- Optimize data-use patterns.

Specialization

Floating Point Arithmetic	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1pJ
32 bit	4pJ

Memory	
Cache	(64 bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3 - 2.6nJ



Data assume a 45 nm process @0.9 V

M. Horowitz, *Computing's energy problem (and what we can do about it)*, IEEE, March. 6, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6757323

Limits to Specialization

- There are costs associated with designing each new accelerator.
- The chip, or “ASIC,” produced may only be competitive in a smaller target market, reducing profitability.
- Specialization reduces flexibility.
  - The logic invested in specialized accelerators is no longer general-purpose.
  - Algorithm changes may render specialized hardware obsolete.
- Once we’ve specialized, further gains may be difficult to achieve.
  - Specialization isn’t immune to the concept of diminishing returns.