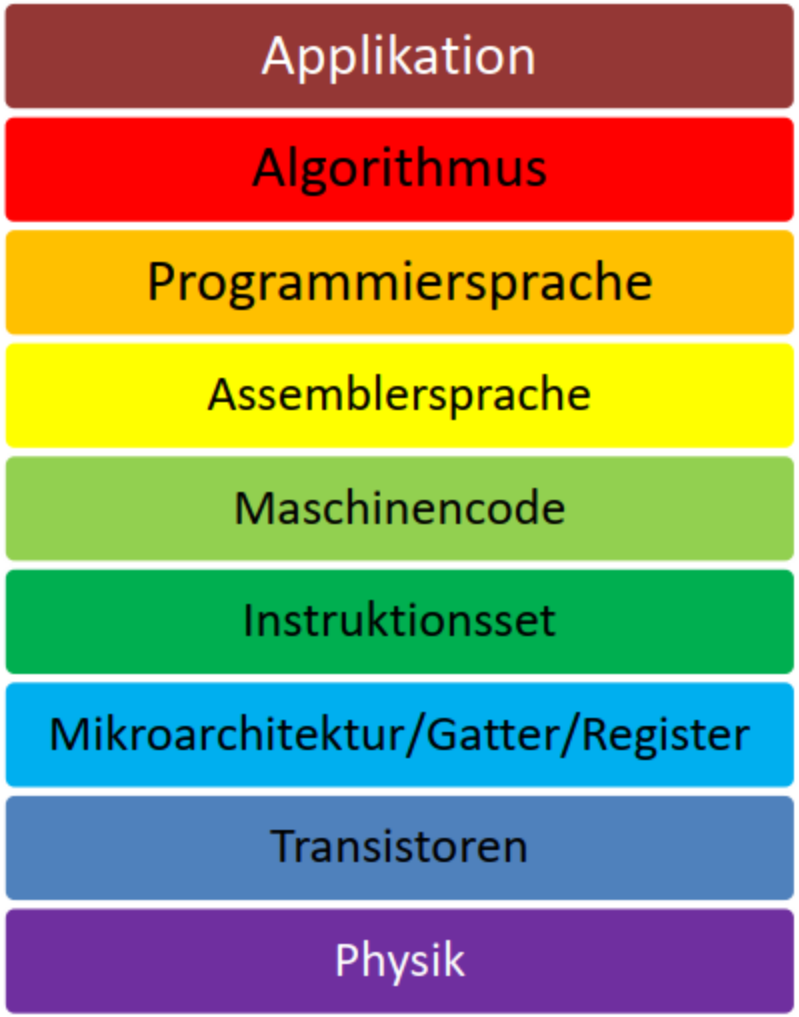
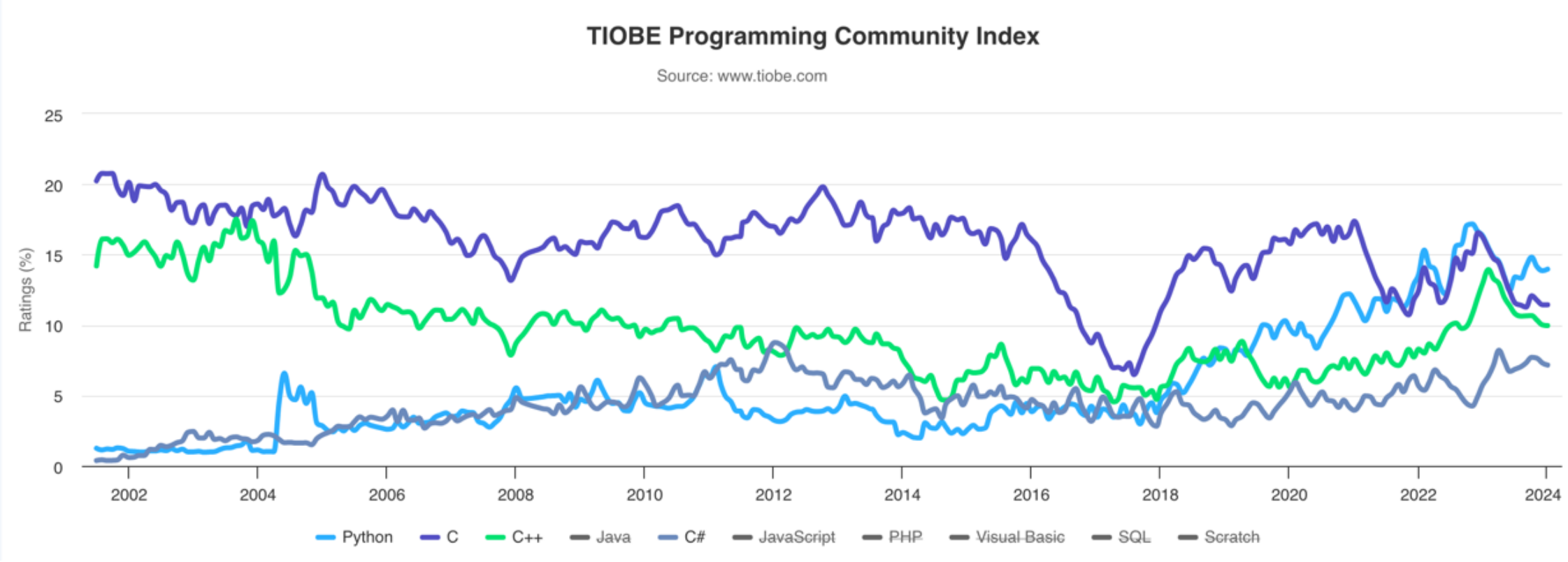
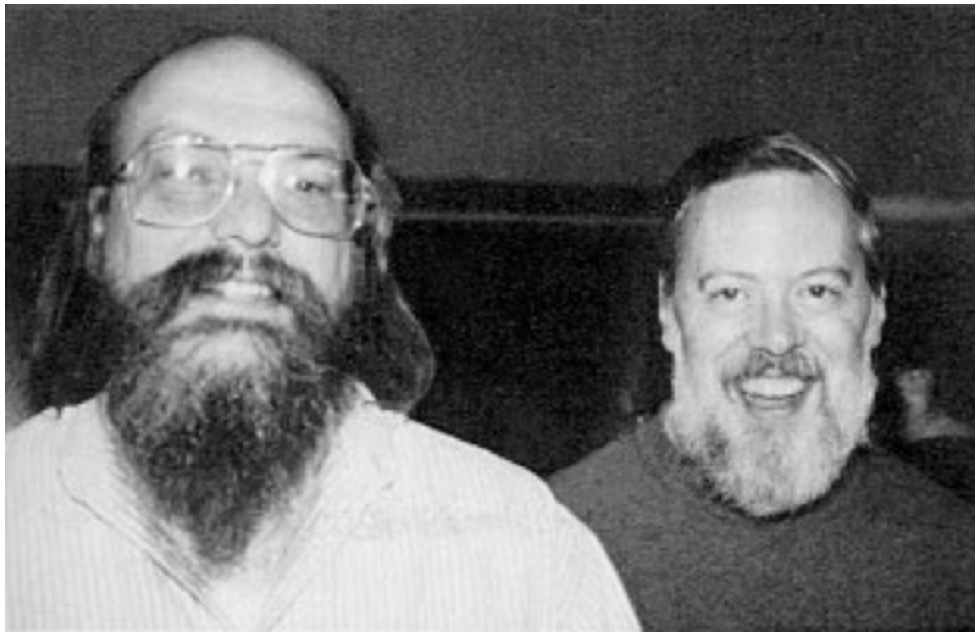
Microcomputertechnik

Überblick


Applikation
Algorithmus
Programmiersprache
Assemblersprache
Maschinencode
Instruktionsset
Mikroarchitektur/Gatter/Register
Transistoren
Physik

Software


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
Python
C
C++
Java
C#
JavaScript
PHP
Visual Basic
SQL
Scratch



Ken Thompson, Dennis Ritchie

C Keywords (Auswahl)

```
bool (C23)   extern       sizeof         default     return
false (C23)  float        static         do          volatile
break        for          struct         double      short
case         goto         switch         else        signed
char         if           true (C23)     unsigned    register
const        int          typedef        void        union
continue     long
```

Python Keywords

```
False       await       else        import      pass
None        break       except      in          raise
True        class       finally     is          return
and         continue    for         lambda      try
as          def         from        nonlocal    while
assert      del         global      not         with
async       elif        if          or          yield
```

Go Keywords

```
break        default      func         interface    select
case         defer        go           map          struct
chan         else         goto         package      switch
const        fallthrough  if           range        type
continue     for          import       return       var
```

https://go.dev/ref/spec#Keywords

Hochsprache zu Maschinencode

Rust

```rust
pub fn square(num: i32) -> i32 {
    num * num
}
```

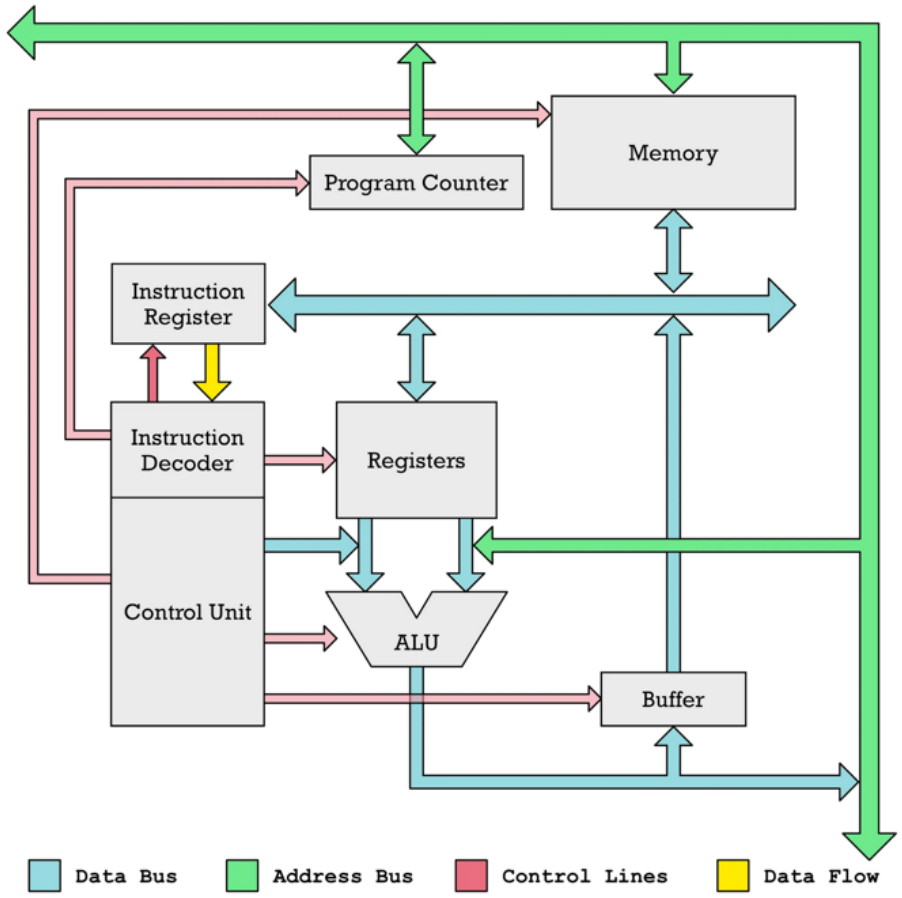
Assembler

```
square:
        push    {r7, lr}
        sub     sp, #8
        smull   r1, r0, r0, r0
        mov     r2, r1
        str     r2, [sp, #4]
        cmp.w   r0, r1, asr #31
        bne     .LBB0_2
        b       .LBB0_1
.LBB0_1:
        ldr     r0, [sp, #4]
        add     sp, #8
        pop     {r7, pc}
.LBB0_2:
        ldr     r0, .LCPI0_0
.LPC0_0:
        add     r0, pc
        ldr     r2, .LCPI0_1
.LPC0_1:
        add     r2, pc
        movs    r1, #33
        bl      core::panicking::panic
        .inst.n 0xdefe
.LCPI0_0:
        .long   str.0-(.LPC0_0+4)
.LCPI0_1:
        .long   .L__unnamed_1-(.LPC0_1+4)
.L__unnamed_2:
        .ascii  "/app/example.rs"
.L__unnamed_1:
        .long   .L__unnamed_2
        .asciz  "\017\000\000\000\013\000\000\000\005\000\000"

str.0:
        .ascii  "attempt to multiply with overflow"
```

godbolt.org

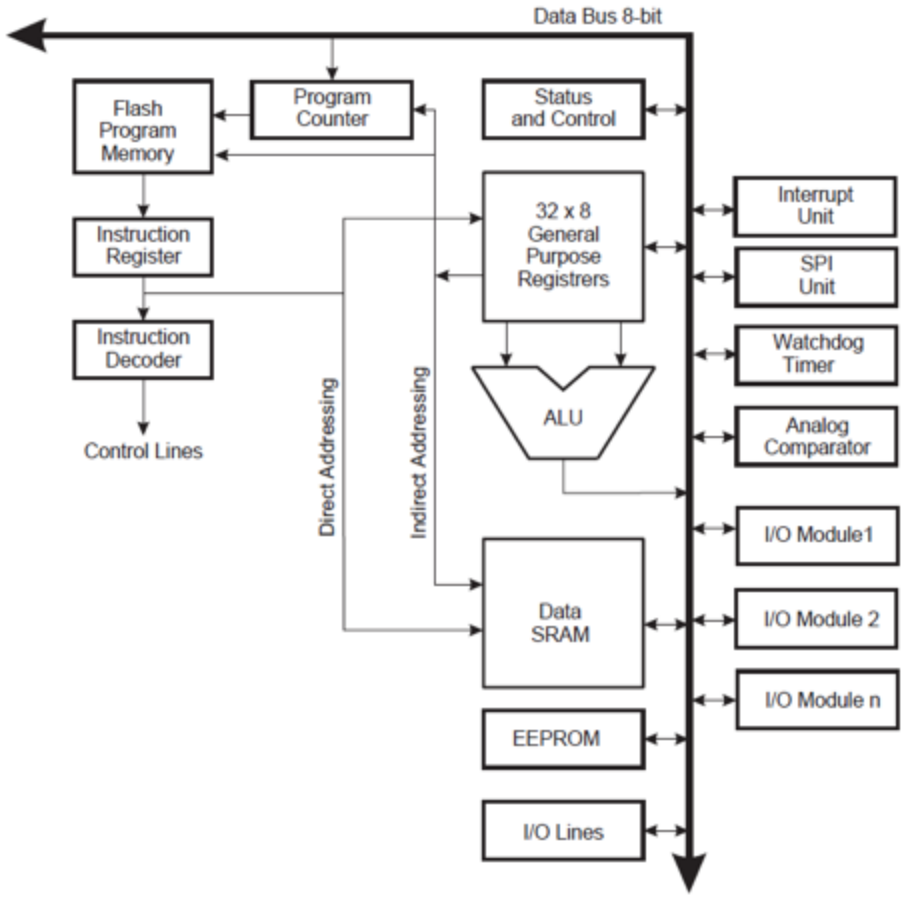
Aufbau und Funktion eines Microprozessors



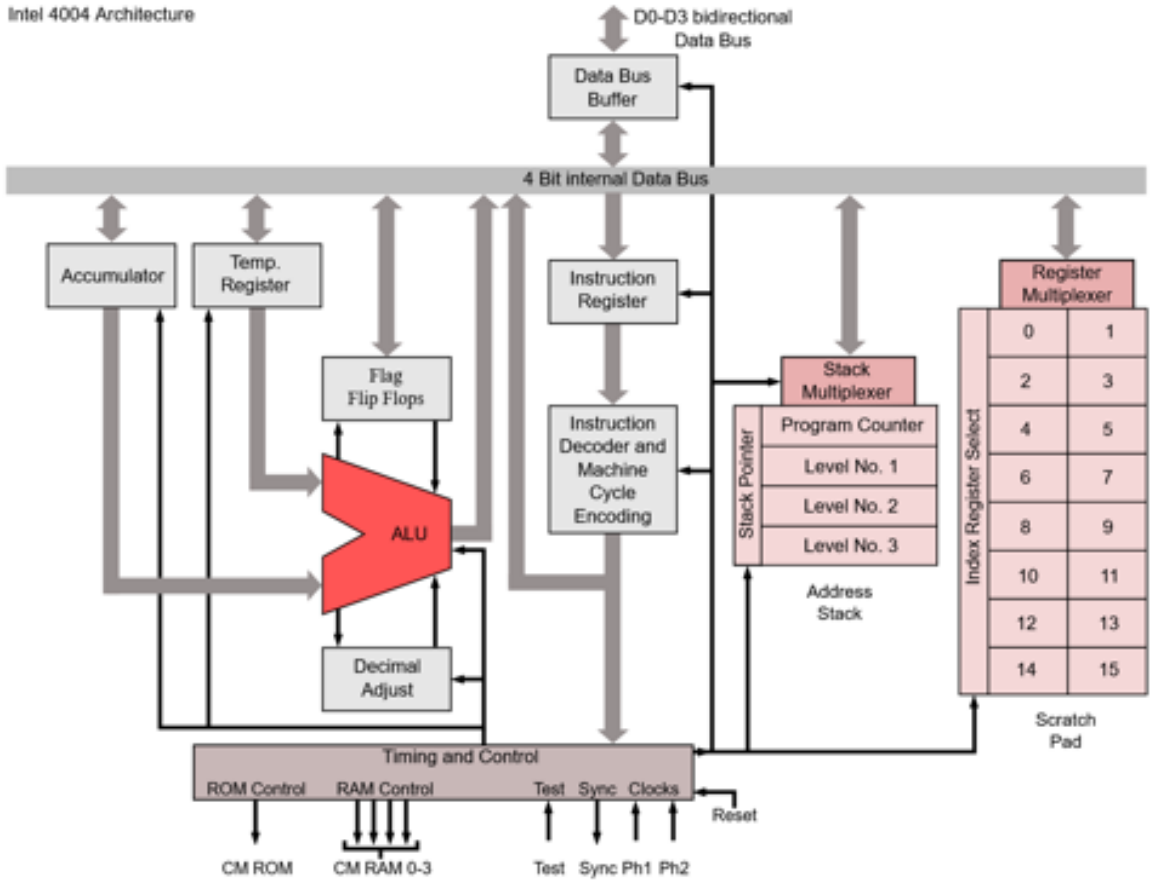
Decoding instruction located in instruction register.

https://erik-engheim.medium.com/how-does-a-microprocessor-run-a-program-11744ab47d04

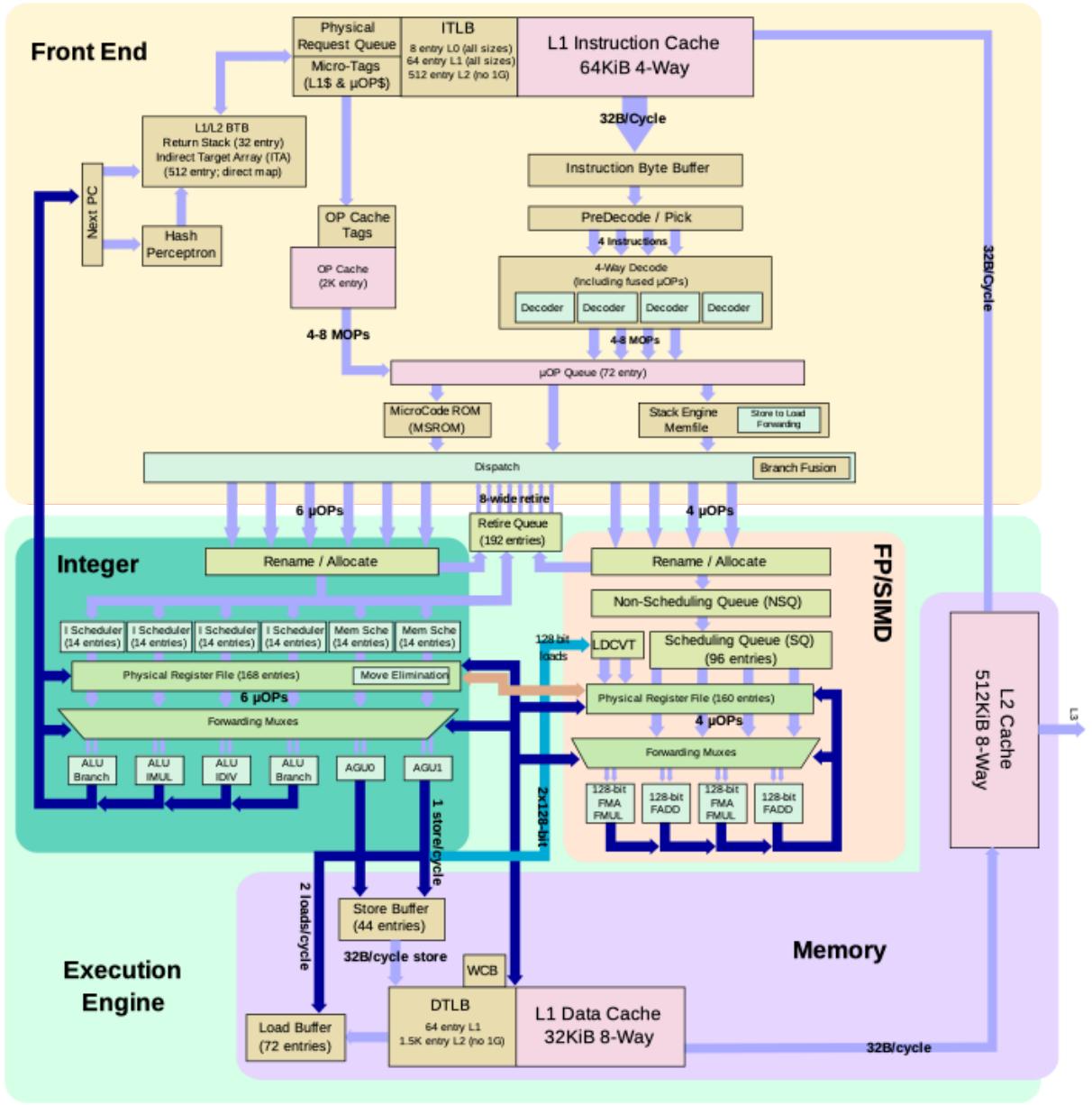
AVR Architektur Blockschaltbild



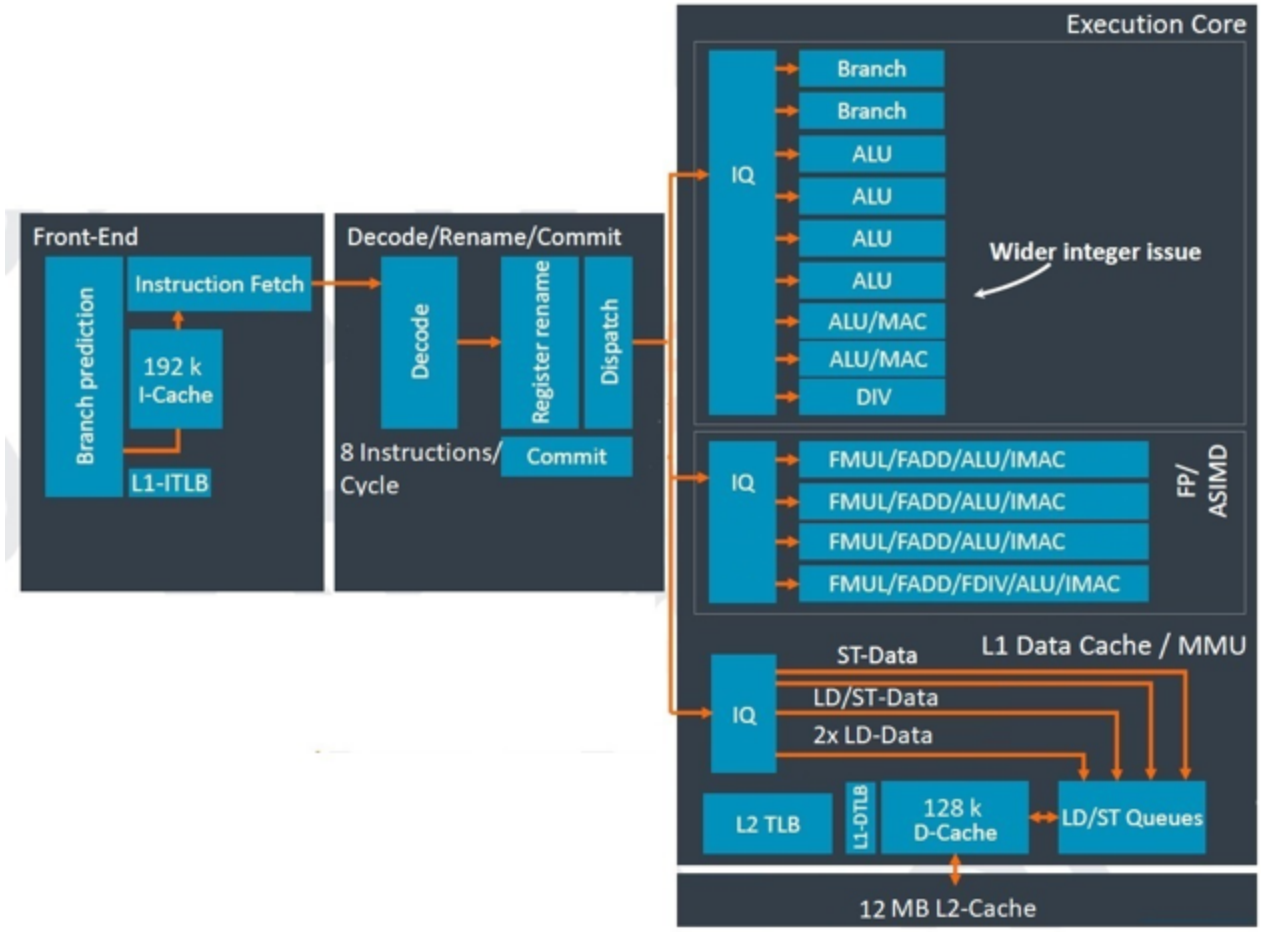
1971: Intel 4004



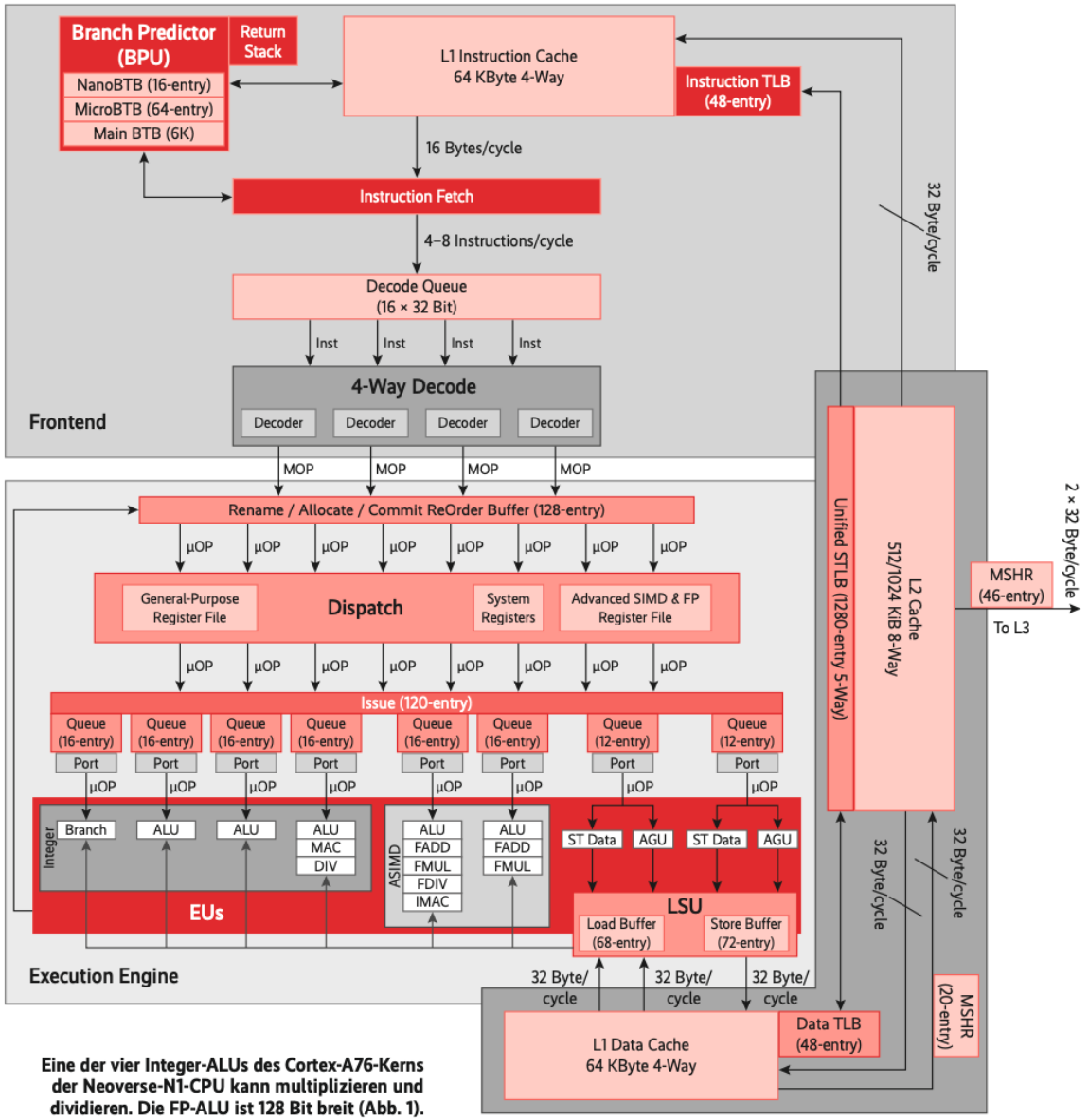
AMD Threadripper



Apple M1

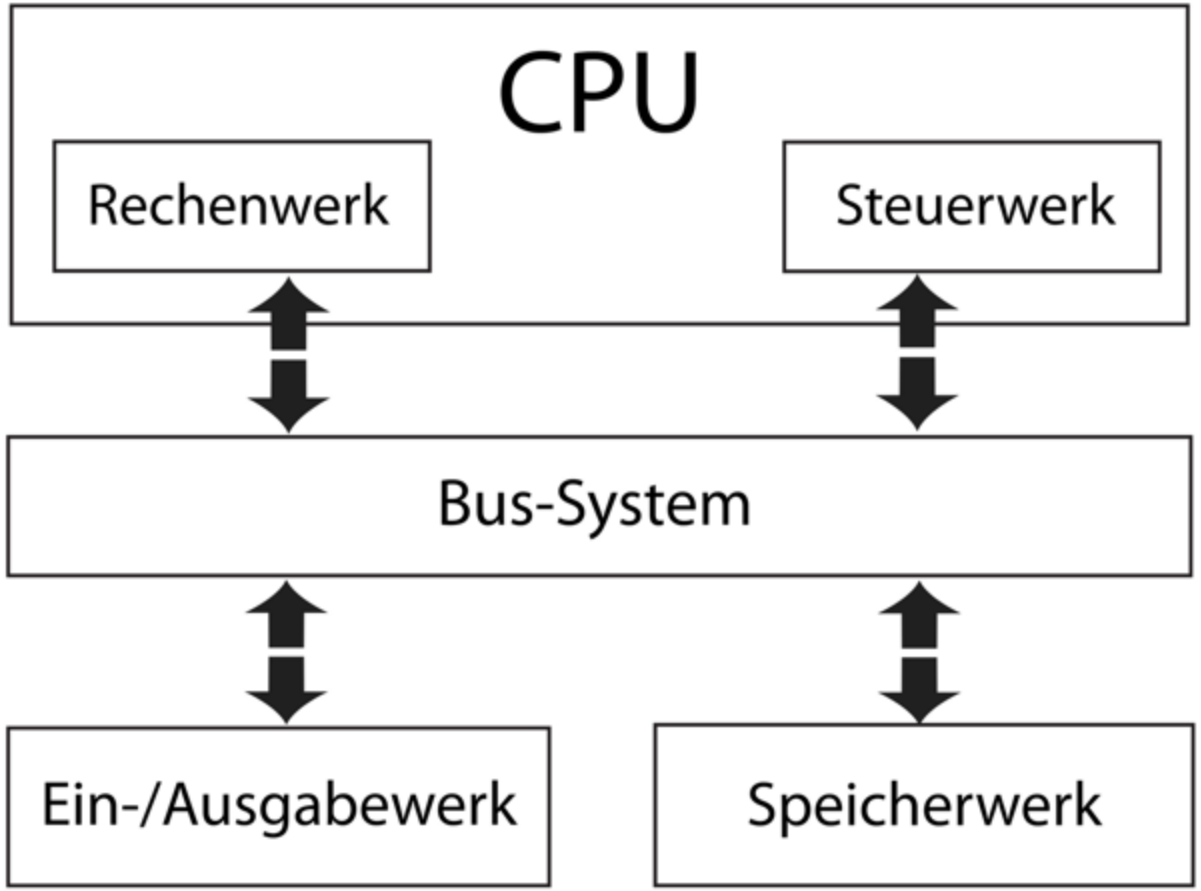

Front-End
Branch prediction
Instruction Fetch
192 k
I-Cache
L1-ITLB
Decode/Rename/Commit
Decode
Register rename
Dispatch
8 Instructions/
Cycle
Commit
Execution Core
IQ
Branch
Branch
ALU
ALU
ALU
ALU
ALU/MAC
ALU/MAC
DIV
Wider integer issue
IQ
FMUL/FADD/ALU/IMAC
FMUL/FADD/ALU/IMAC
FMUL/FADD/ALU/IMAC
FMUL/FADD/FDIV/ALU/IMAC
FP/
ASIMD
L1 Data Cache / MMU
IQ
ST-Data
LD/ST-Data
2x LD-Data
L2 TLB
L1-DTLB
128 k
D-Cache
LD/ST Queues
12 MB L2-Cache

ARM Cortex A67

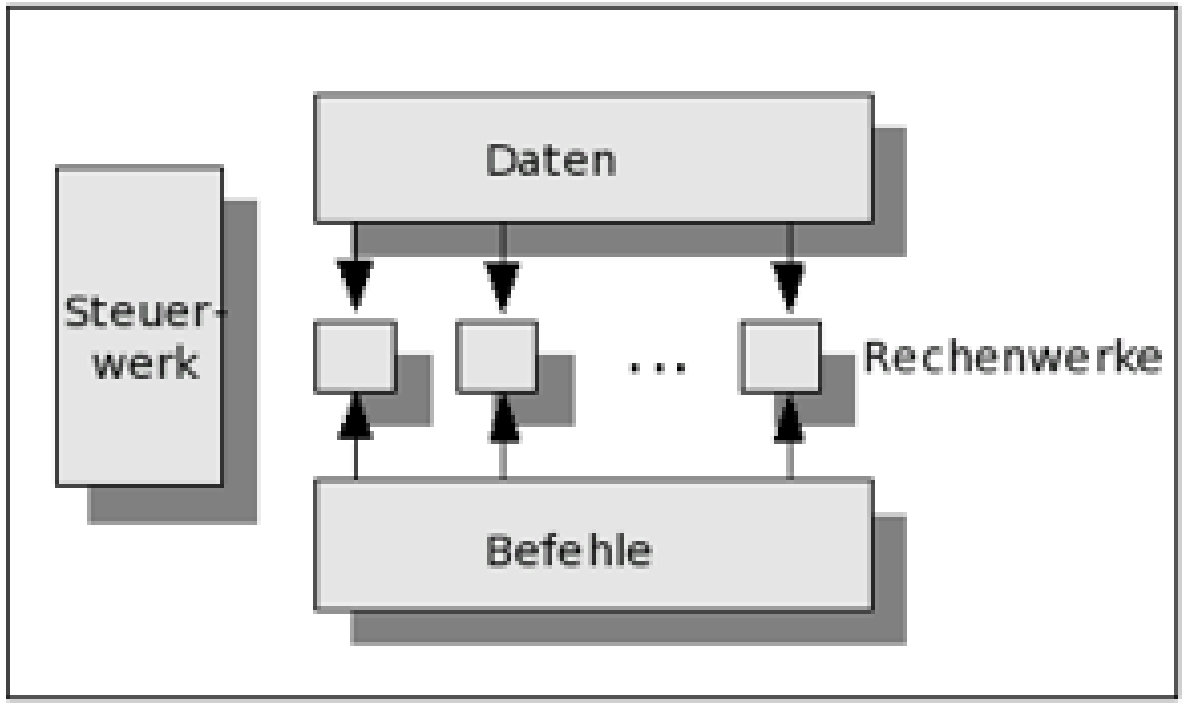


Eine der vier Integer-ALUs des Cortex-A76-Kerns der Neoverse-N1-CPU kann multiplizieren und dividieren. Die FP-ALU ist 128 Bit breit (Abb. 1).

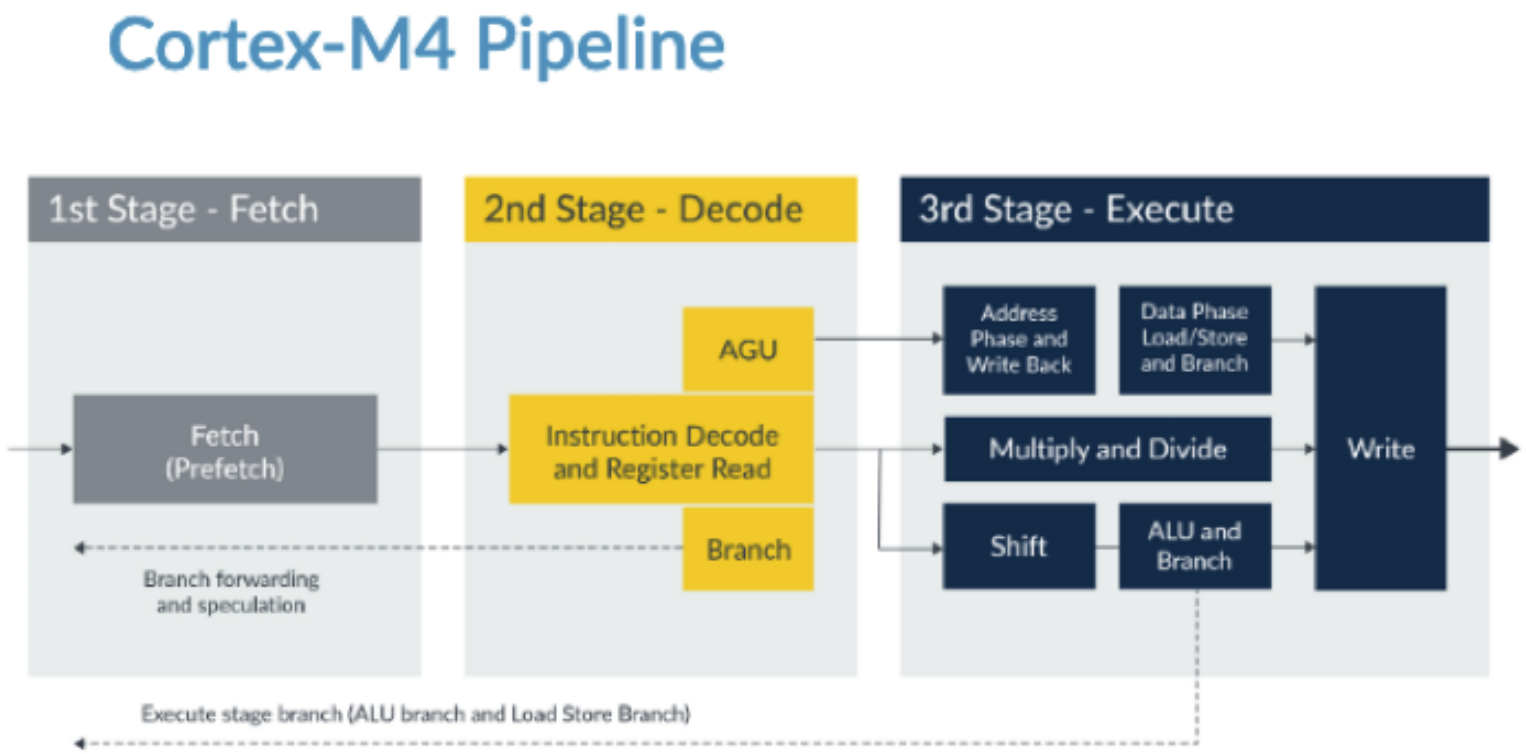
von Neumann Architektur



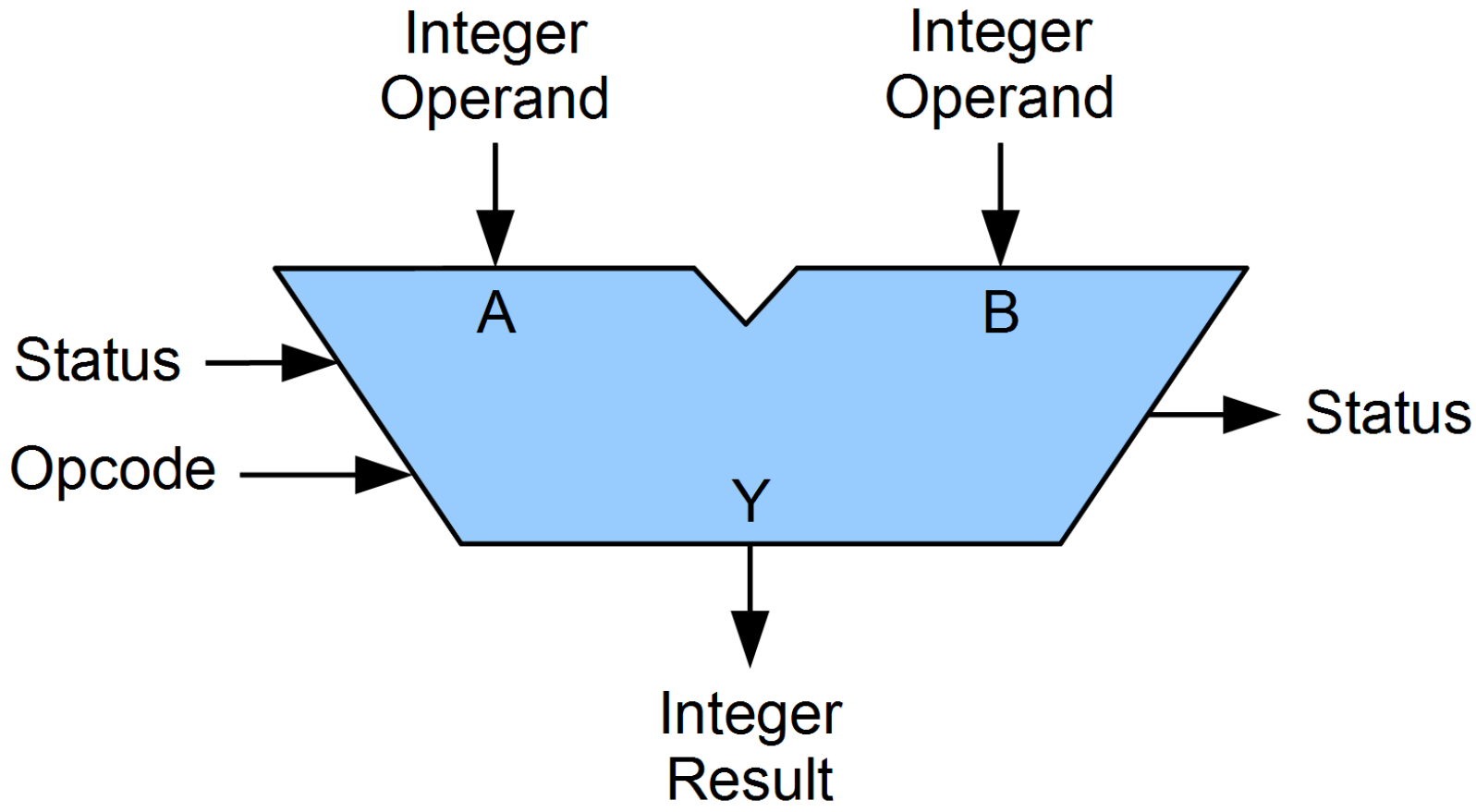
Harvard Architektur



Fetch - Decode - Execute



Arithmetic Logic Unit (ALU)



Mindestens:

- Addition (ADD)
- Negation (NOT)
- Konjunktion (AND)

Zusätzlich (Auswahl):

- Subtraktion
- Vergleich
- Multiplikationen / Division
- Oder
- Shift / Rotation

Instruction Set

http://lyons42.com/AVR/Opcodes/AVRAllOpcodes.html

A64 Instruction Set

C4.1 A64 instruction set encoding

The A64 instruction encoding is:

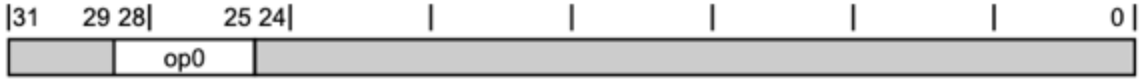
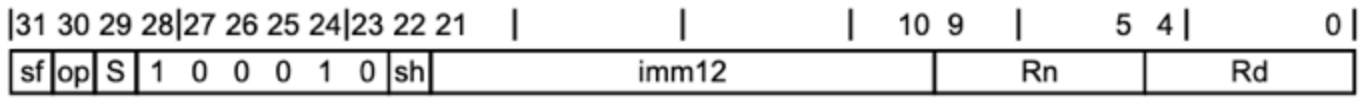


Table C4-1 Main encoding table for the A64 instruction set

Decode fields op0	Decode group or instruction page
0000	*Reserved* on page C4-284.
0001	Unallocated.
0010	SVE instructions. See *The Scalable Vector Extension (SVE)* on page A2-110.
0011	Unallocated.
100x	*Data Processing -- Immediate* on page C4-284.
101x	*Branches, Exception Generating and System instructions* on page C4-289.
x1x0	*Loads and Stores* on page C4-298.
x101	*Data Processing -- Register* on page C4-332.
x111	*Data Processing -- Scalar Floating-Point and Advanced SIMD* on page C4-342.

Table C4-3 Encoding table for the Data Processing -- Immediate group

Decode fields op0	Decode group or instruction page
00x	*PC-rel. addressing* on page C4-285
010	*Add/subtract (immediate)* on page C4-285
011	*Add/subtract (immediate, with tags)* on page C4-286
100	*Logical (immediate)* on page C4-286
101	*Move wide (immediate)* on page C4-287
110	*Bitfield* on page C4-288
111	*Extract* on page C4-288



Decode fields			Instruction page
sf	op	S	
0	0	0	ADD (immediate) - 32-bit variant
0	0	1	ADDS (immediate) - 32-bit variant
0	1	0	SUB (immediate) - 32-bit variant
0	1	1	SUBS (immediate) - 32-bit variant
1	0	0	ADD (immediate) - 64-bit variant
1	0	1	ADDS (immediate) - 64-bit variant
1	1	0	SUB (immediate) - 64-bit variant
1	1	1	SUBS (immediate) - 64-bit variant

Instruction Set

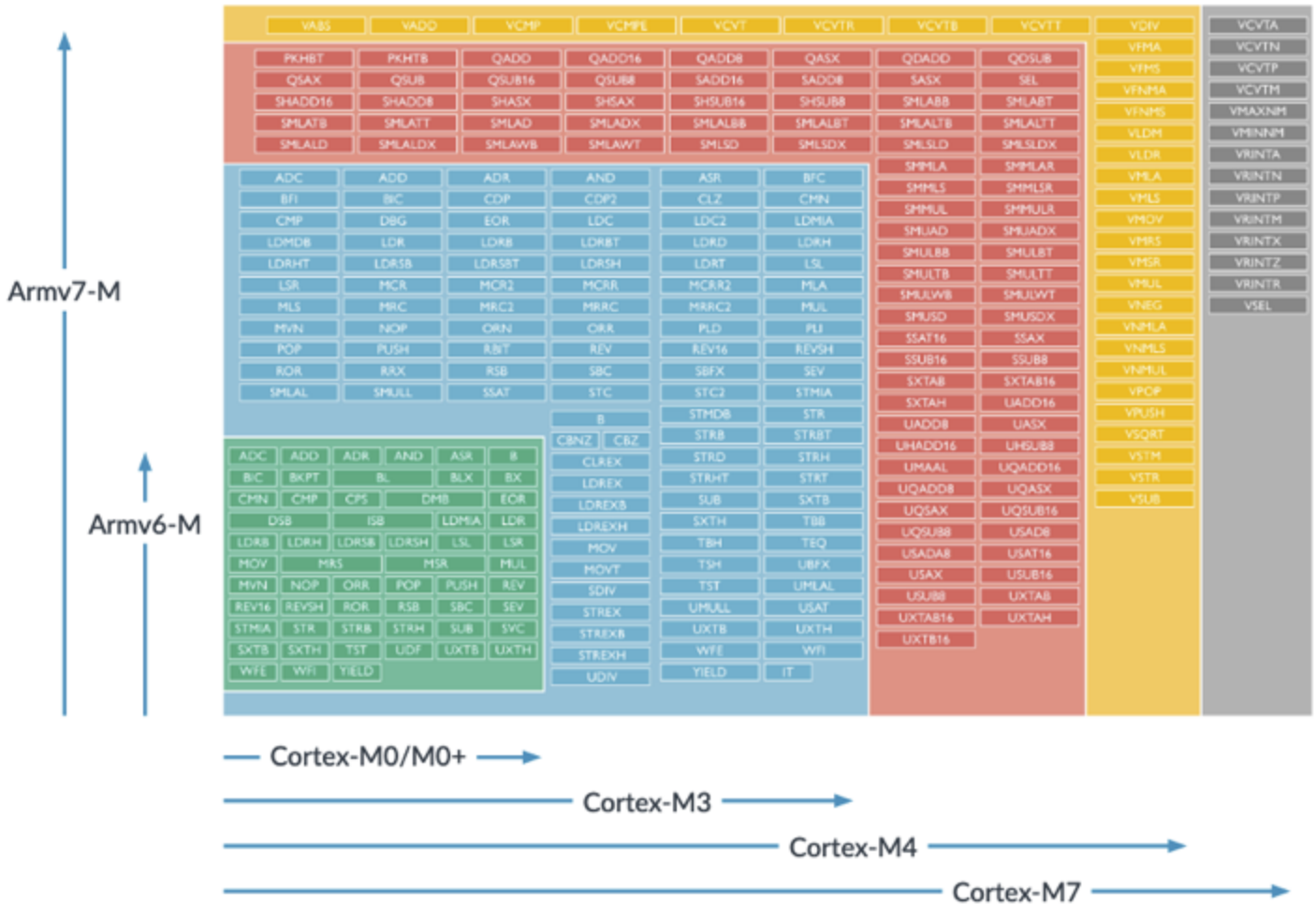


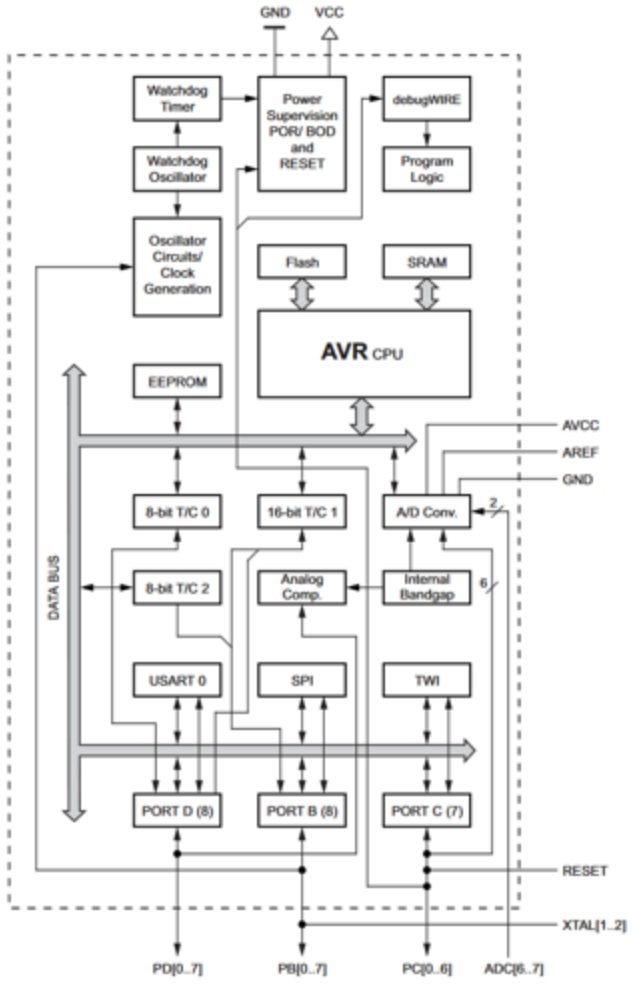
Figure 5: Instruction set

Reduced Instruction Set Computer (RISC)

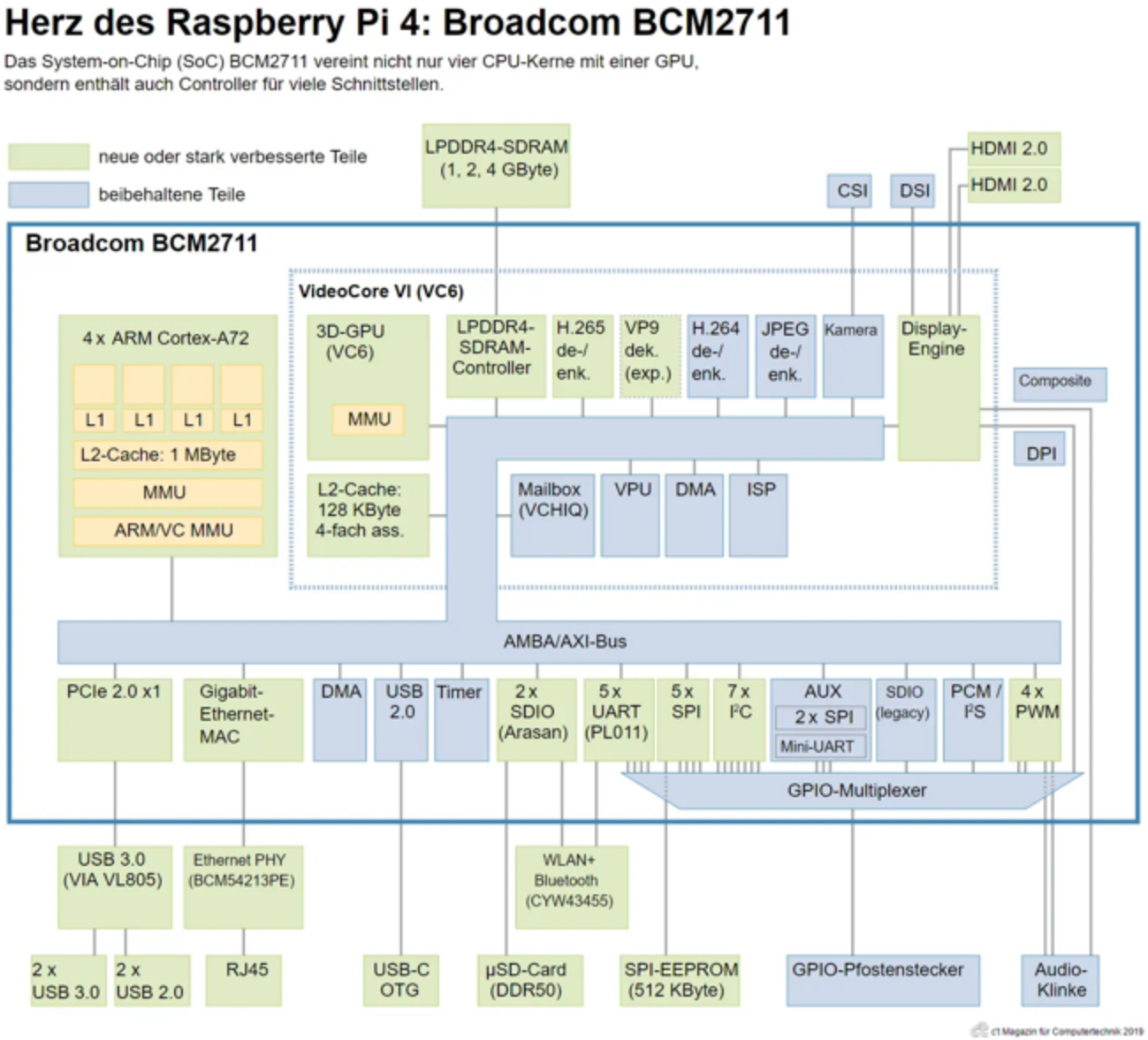
- Opcode hat eine feste Länge
- Meistens 1 Takt pro Operation
- Load/Store Architektur: Separate Lade und Speicher-Befehle
- Hohe Anzahl an Registern für Zwischenresultate
- Oft Harvard-Architektur
- Grundsätzlich: Einfachere Architektur, einfacher für Compiler
- Alles andere: **CISC**

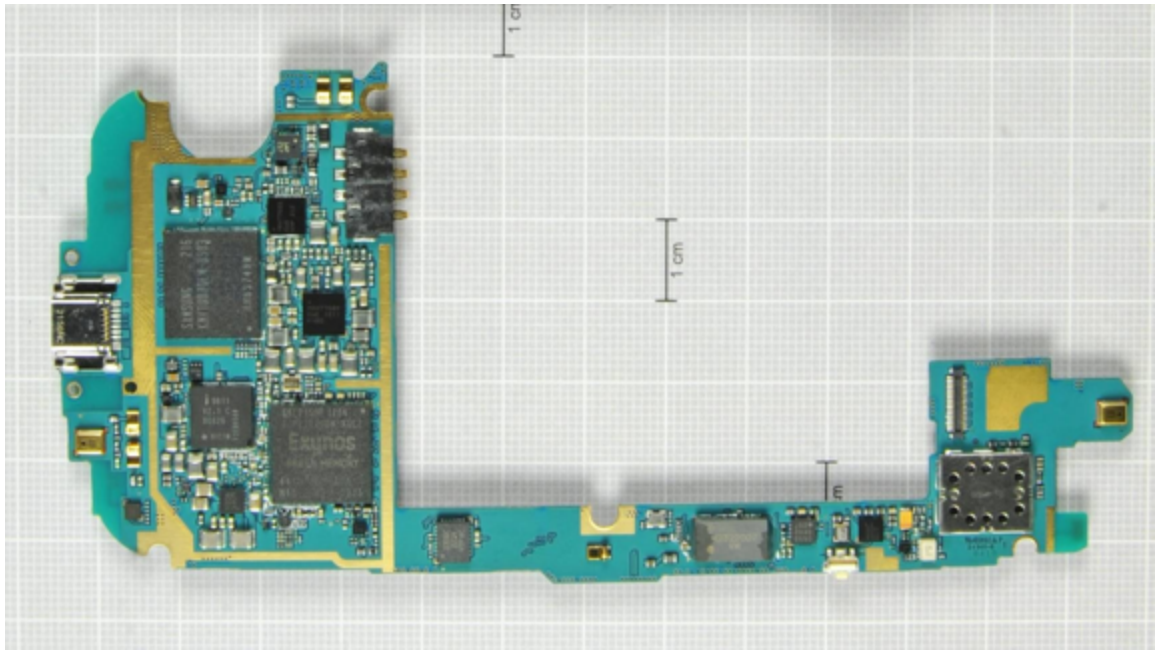
SoC vs Microprocessor vs Microcontroller

Microcontroller: ATmega328P

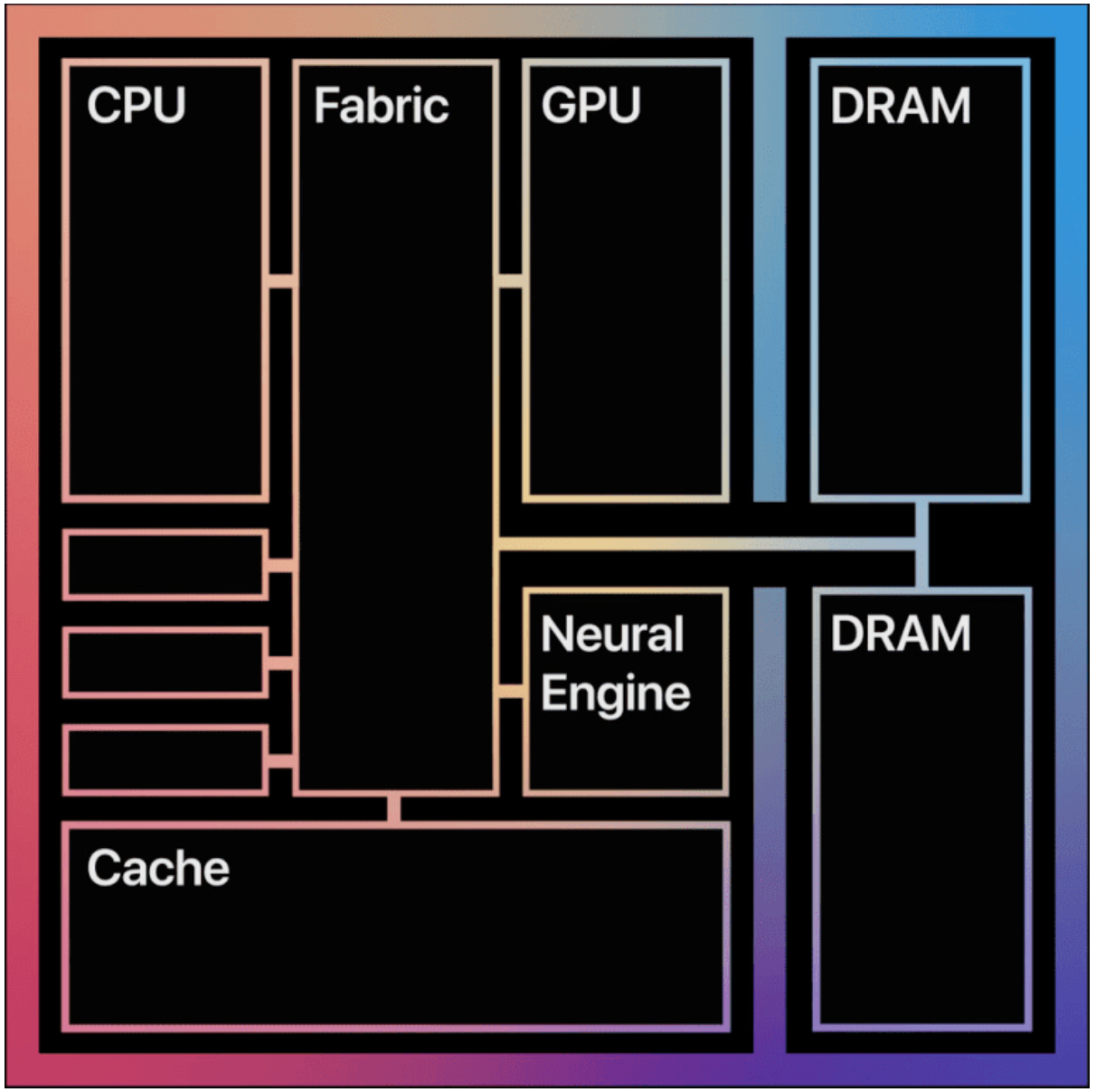


System on Chip (SoC)



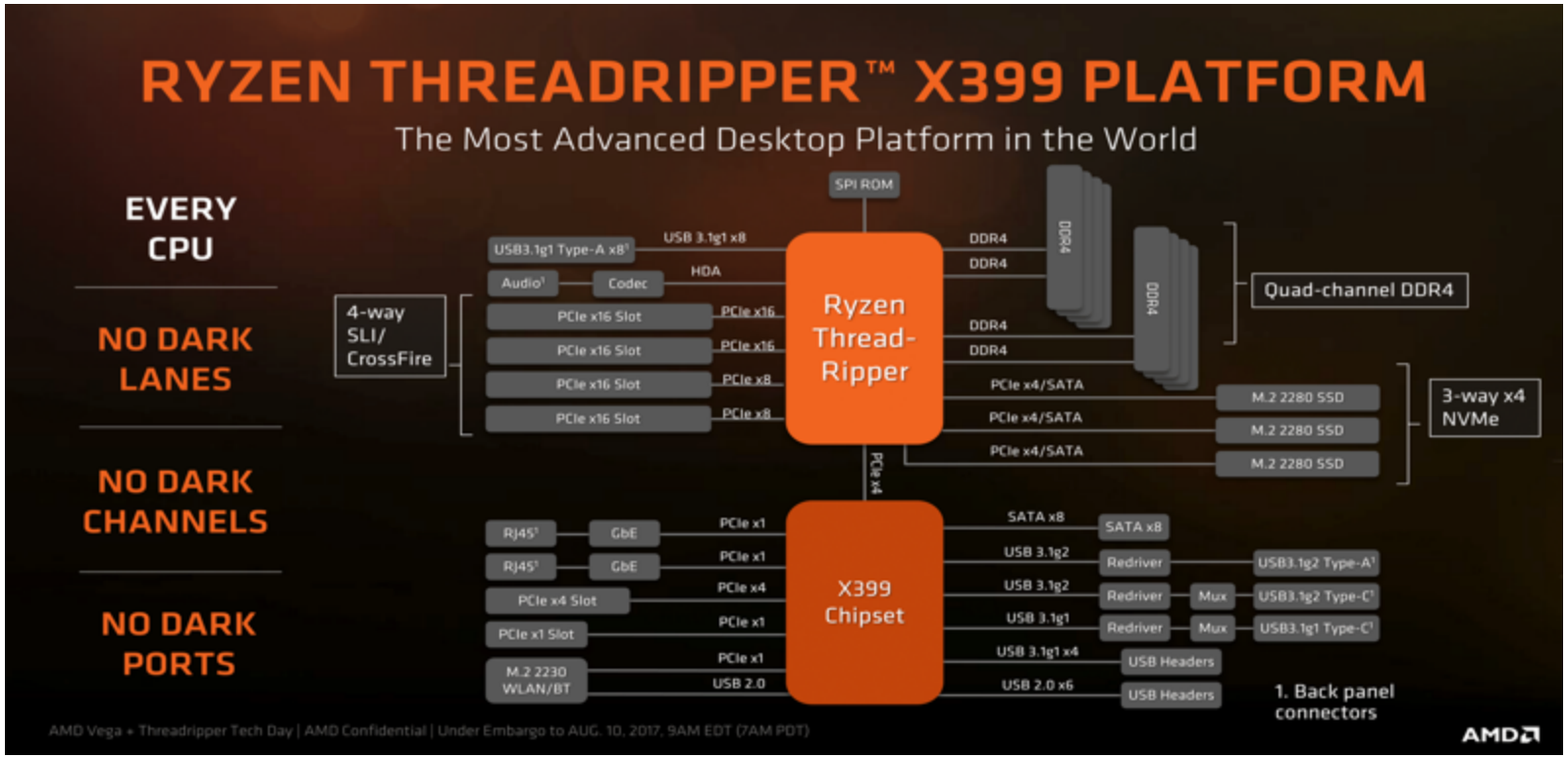


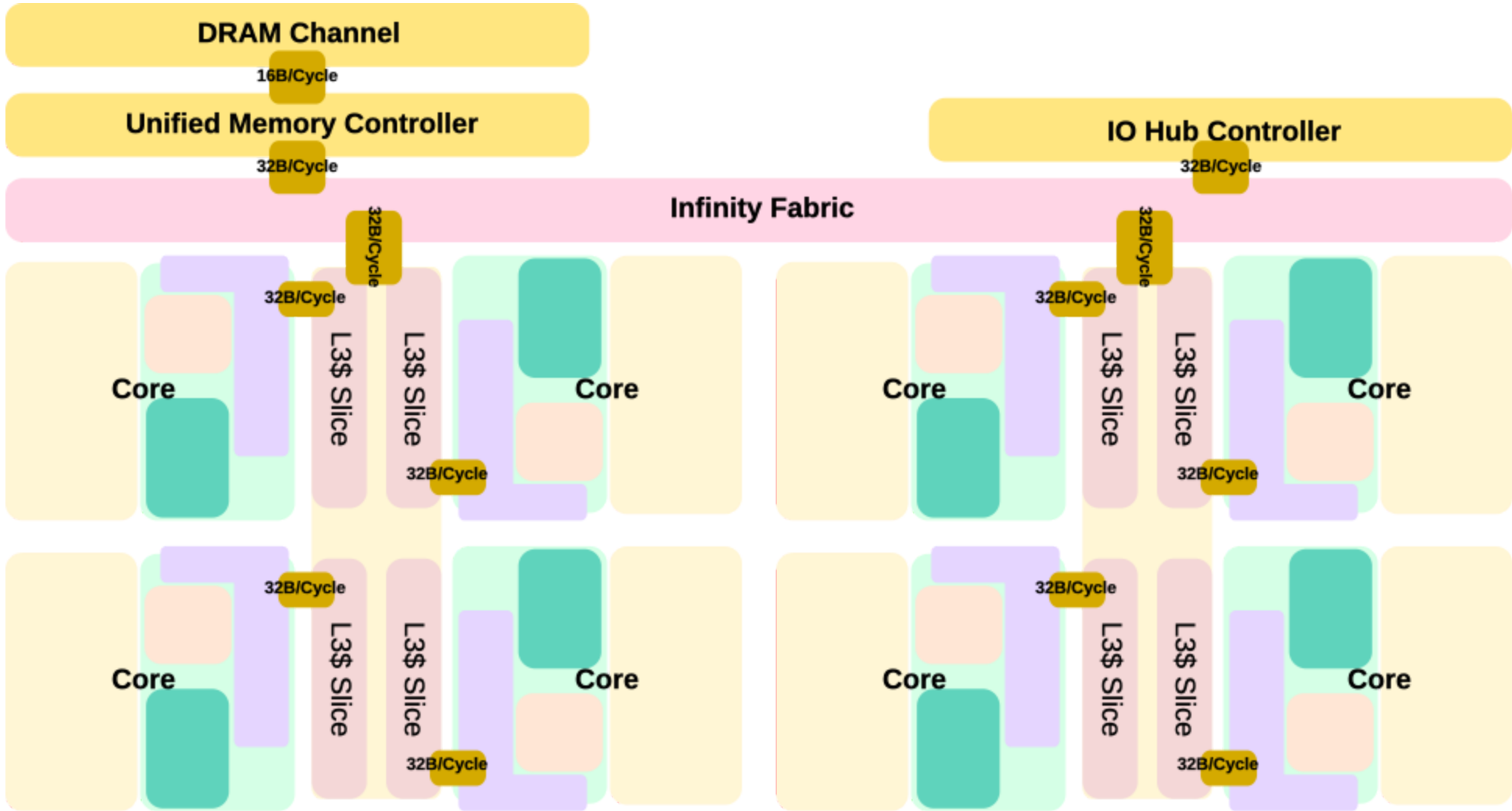
Samsung Galaxy S3



Apple M1

Microprocessor: AMD Ryzen Threadripper




DRAM Channel
16B/Cycle
Unified Memory Controller
32B/Cycle
IO Hub Controller
32B/Cycle
Infinity Fabric
32B/Cycle
32B/Cycle
L3$ Slice
L3$ Slice
32B/Cycle
Core
Core
32B/Cycle
32B/Cycle
L3$ Slice
L3$ Slice
32B/Cycle
Core
Core
32B/Cycle
32B/Cycle
L3$ Slice
L3$ Slice
32B/Cycle
Core
Core
32B/Cycle
32B/Cycle
L3$ Slice
L3$ Slice
32B/Cycle
Core
Core

Advanced RISC Machine (ARM)

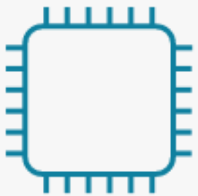
> "Arm licenses processor designs to semiconductor companies that incorporate the technology into their computer chips.
> Licensees pay an up-front fee to gain access to our technology, and a royalty on every chip that uses one of our technology designs.
> Typically, the royalty is based on the selling price of the chip."

(https://group.softbank/en/ir/financials/annual_reports/2021/message/segars, 08.01.2024)

Company Highlights

70%

of the world's population uses Arm-based products

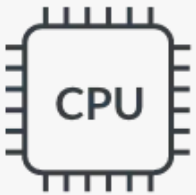


270Bn+

Arm-based chips shipped to date

99%

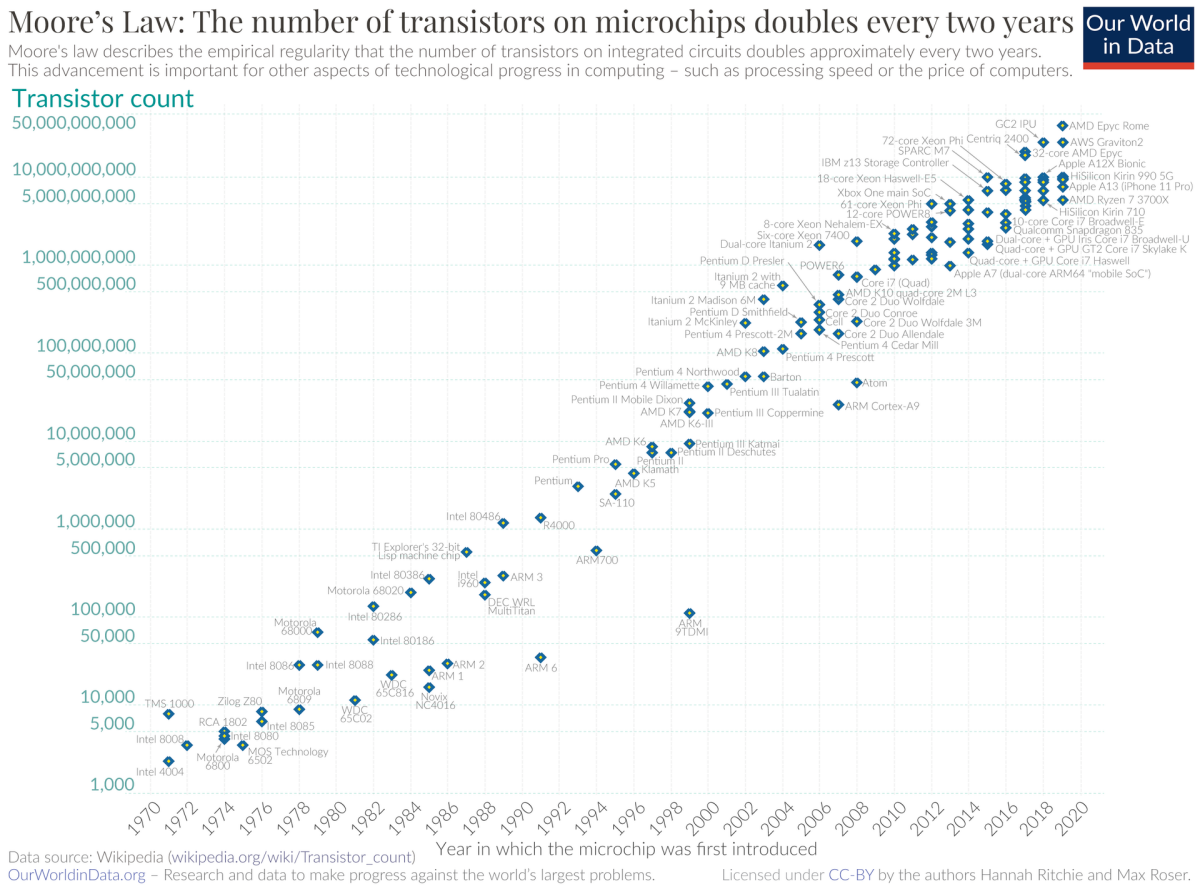
of smartphones run on Arm-based processors

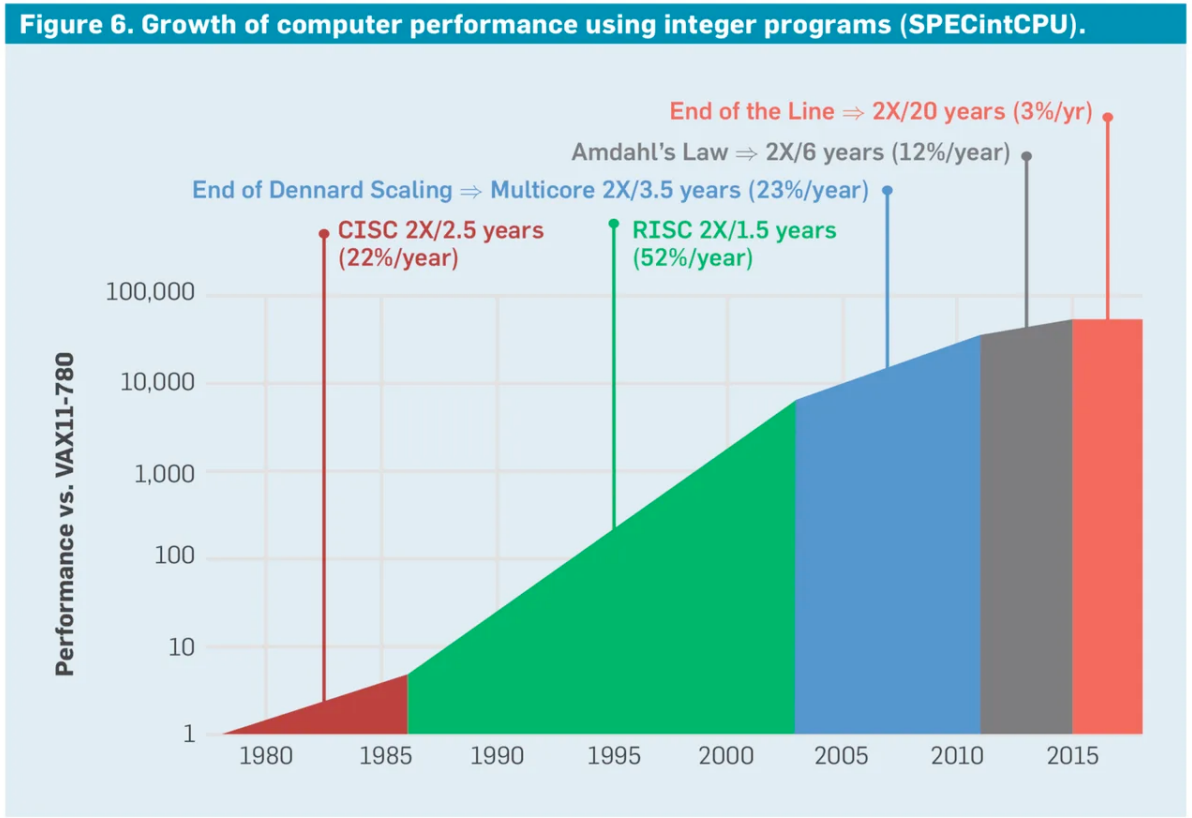


50%

of all chips with processors are Arm-based

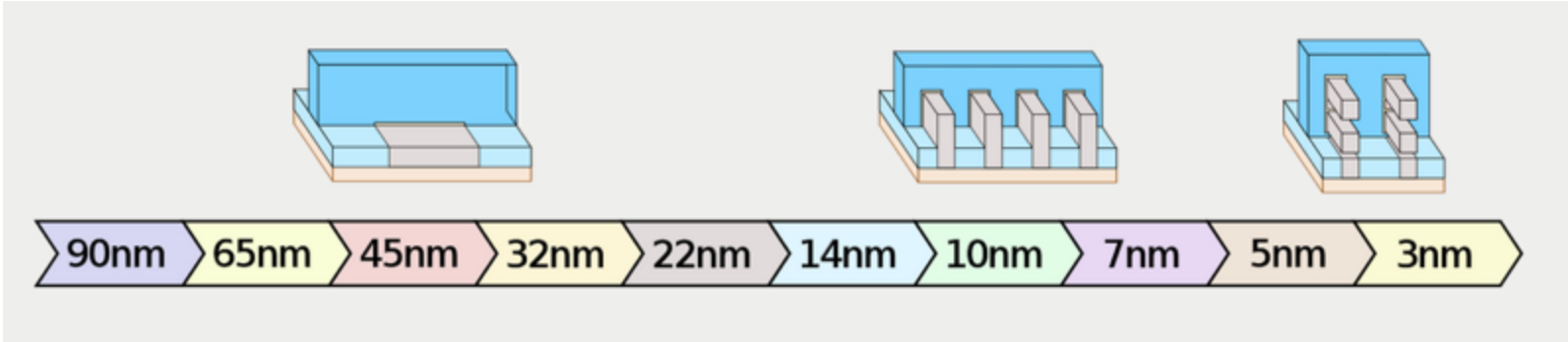
Moore's Law


Moore's Law: The number of transistors on microchips doubles every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Our World in Data
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year in which the microchip was first introduced
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

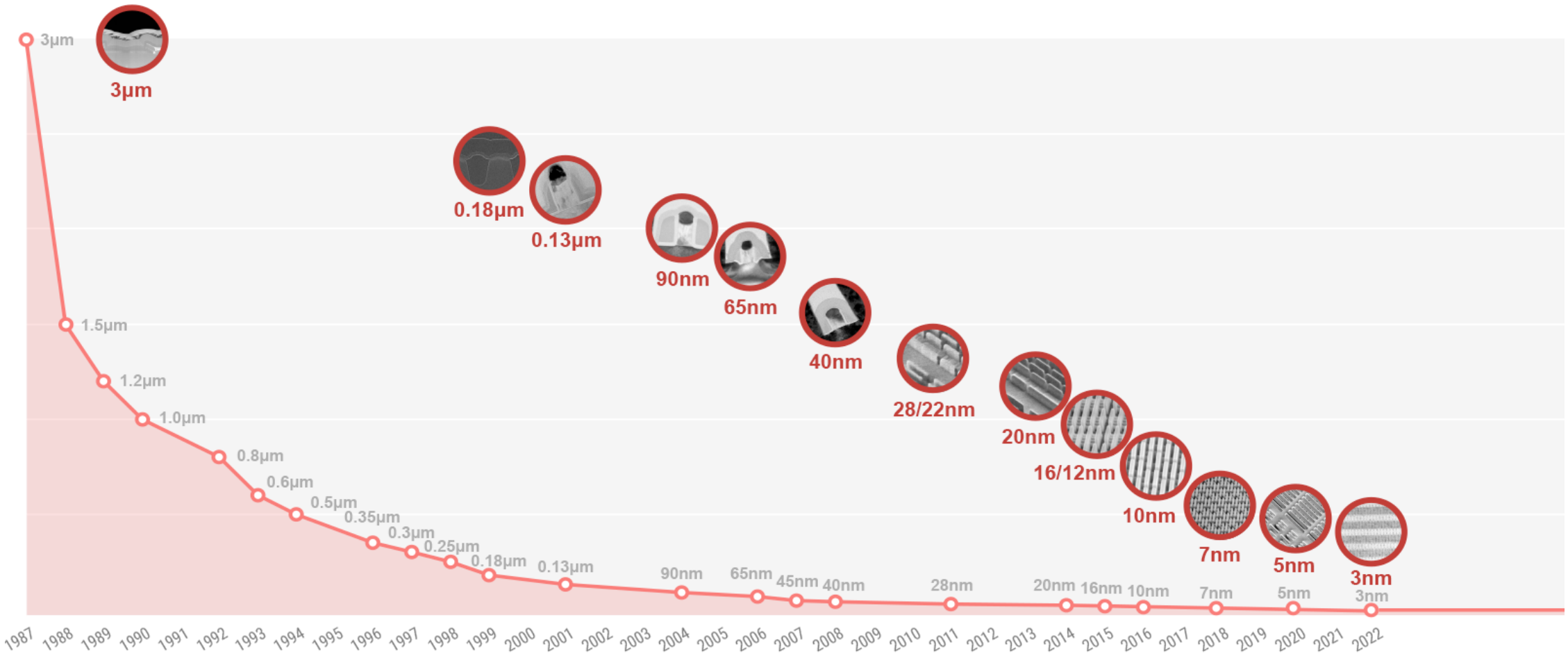


https://www.zdnet.com/article/ai-is-changing-the-entire-nature-of-compute/

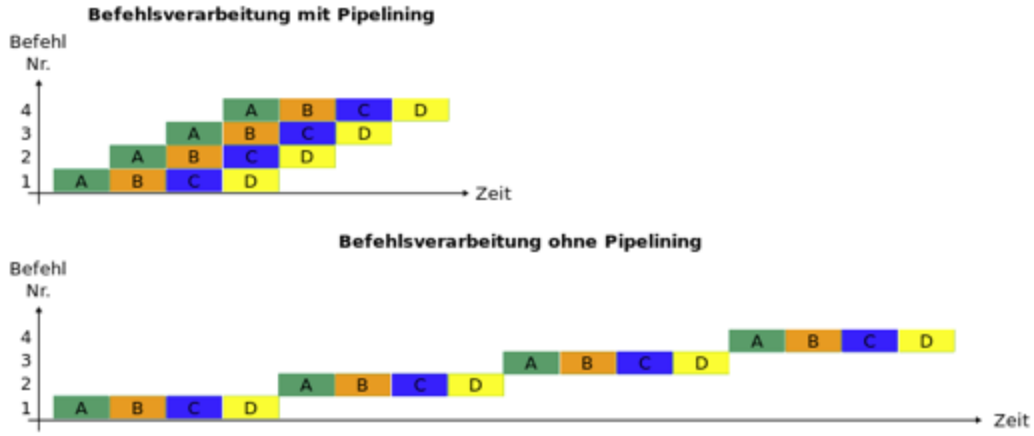
Strukturgrösse



TSMC


3μm
1.5μm
1.2μm
1.0μm
0.8μm
0.6μm
0.5μm
0.35μm
0.3μm
0.25μm
0.18μm
0.13μm
90nm
65nm
45nm
40nm
28nm
20nm
16nm
10nm
7nm
5nm
3nm
3μm
0.18μm
0.13μm
90nm
65nm
40nm
28/22nm
20nm
16/12nm
10nm
7nm
5nm
3nm
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022

Pipelining



A – Befehlscode laden (IF, Instruction Fetch)

In der Befehlsbereitstellungsphase wird der Befehl, der durch den Befehlszähler adressiert ist, aus dem Arbeitsspeicher geladen. Der Befehlszähler wird anschließend hochgezählt.

B – Instruktion dekodieren und Laden der Daten (ID, Instruction Decoding)

In der Dekodier- und Ladephase wird der geladene Befehl dekodiert (1. Takthälfte) und die notwendigen Daten aus dem Arbeitsspeicher und dem Registersatz geladen (2. Takthälfte).

C – Befehl ausführen (EX, Execution)

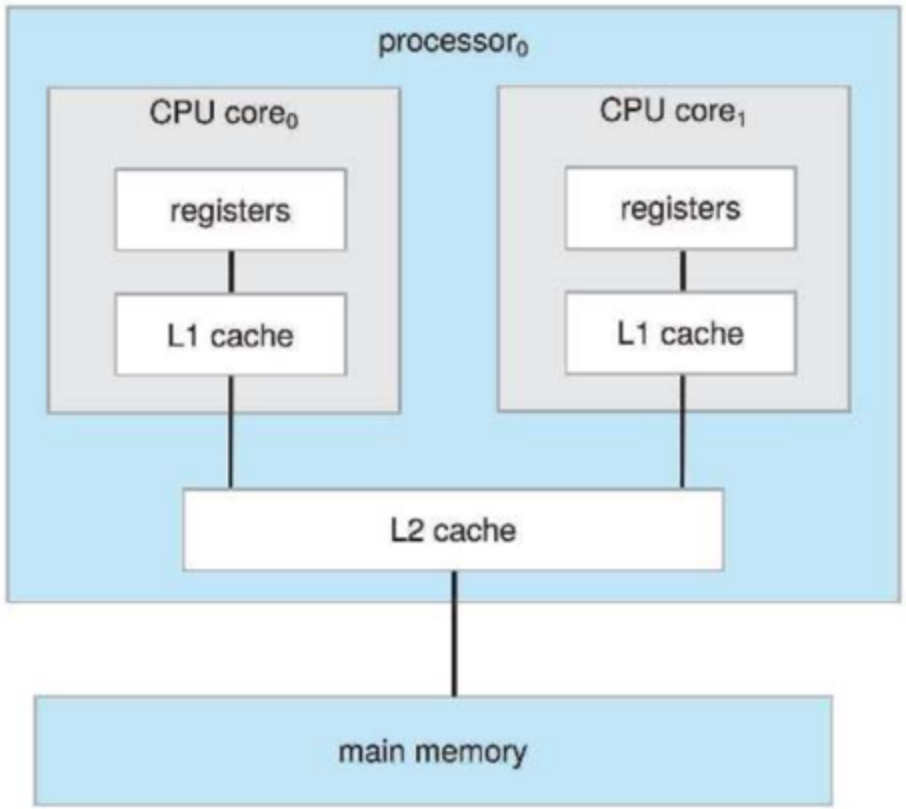
In der Ausführungsphase wird der dekodierte Befehl ausgeführt. Das Ergebnis wird durch den Pipeline-latch gepuffert.

D – Ergebnisse zurückgeben (WB, Write Back)

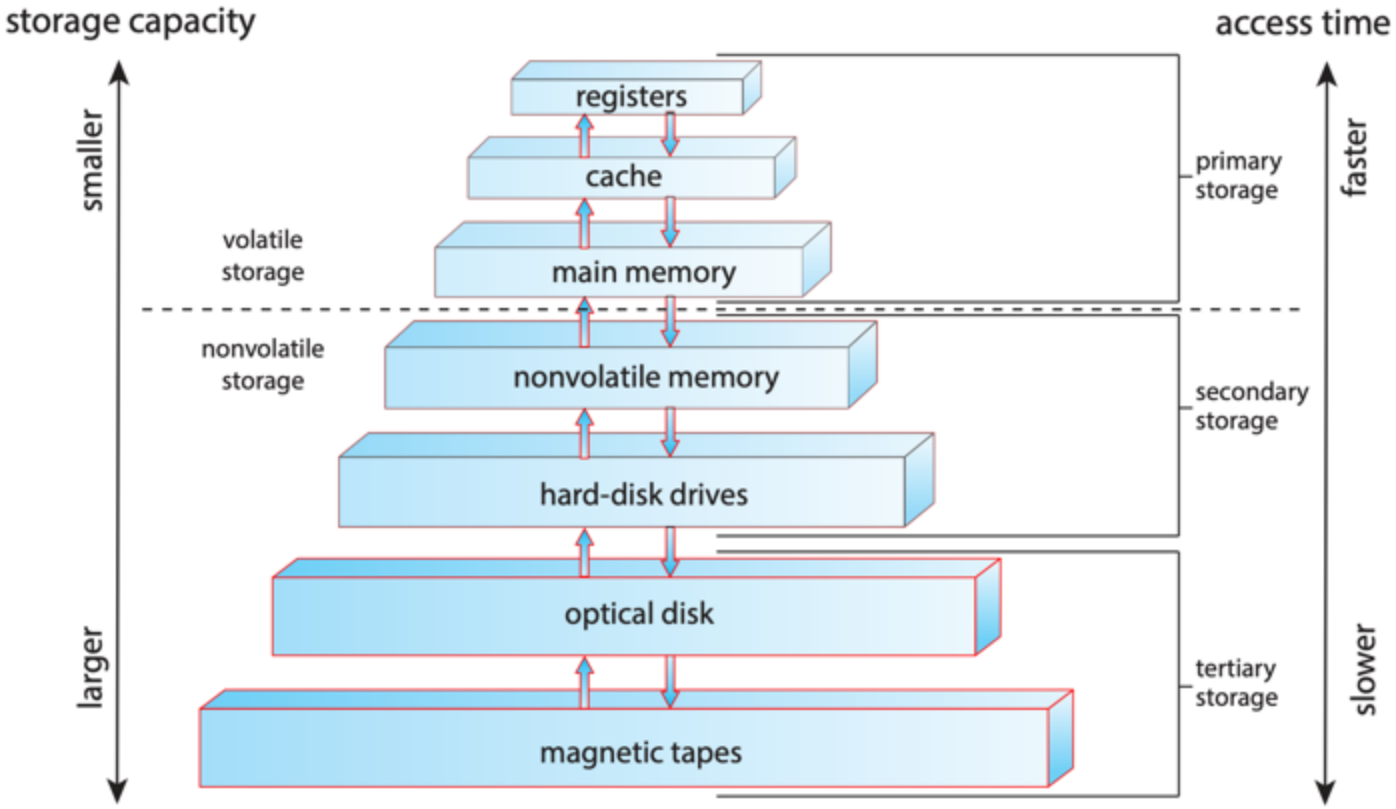
In der Resultatspeicherphase wird das Ergebnis in den Arbeitsspeicher oder in den Registersatz zurückgeschrieben.

Speicher

Cache



(Silberschatz, 2019)



(Silberschatz, 2019)

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Figure 1.14 Characteristics of various types of storage.

(Silberschatz, 2019)

Quellen

Silberschatz, 2019

: A.Silberschatz, P.B.Galvin, G. Gagne (2019): Operating System Concepts, Global Edition, Wiley