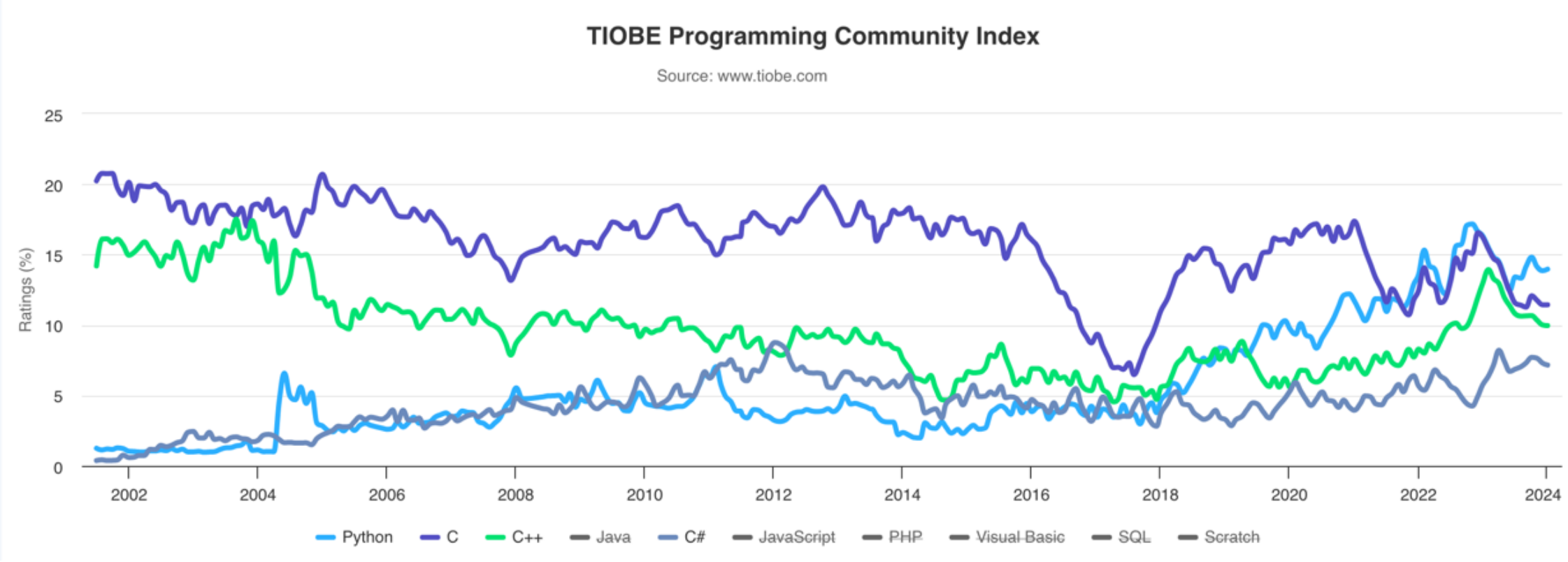
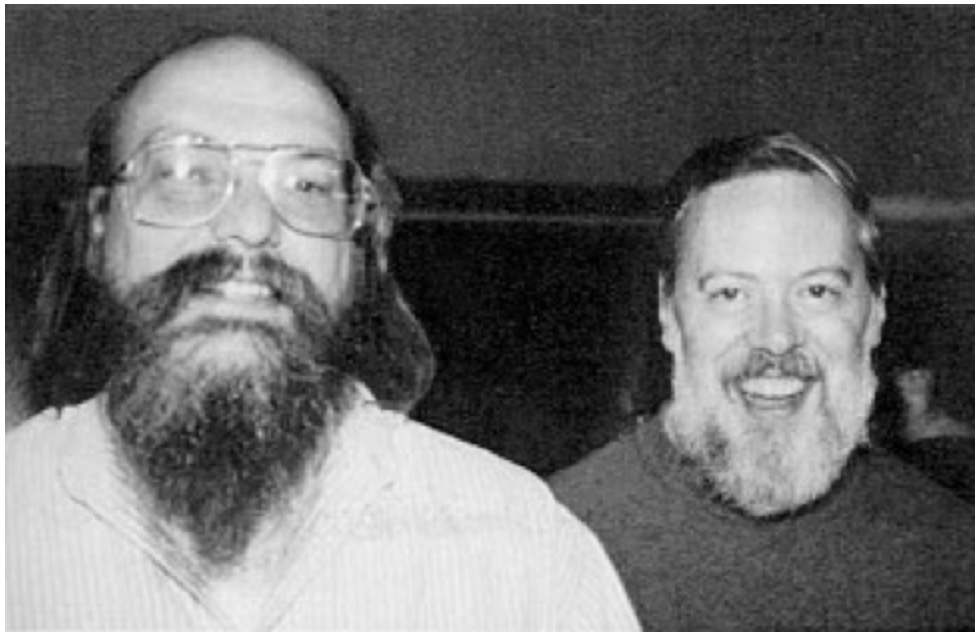
Microcomputertechnik

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

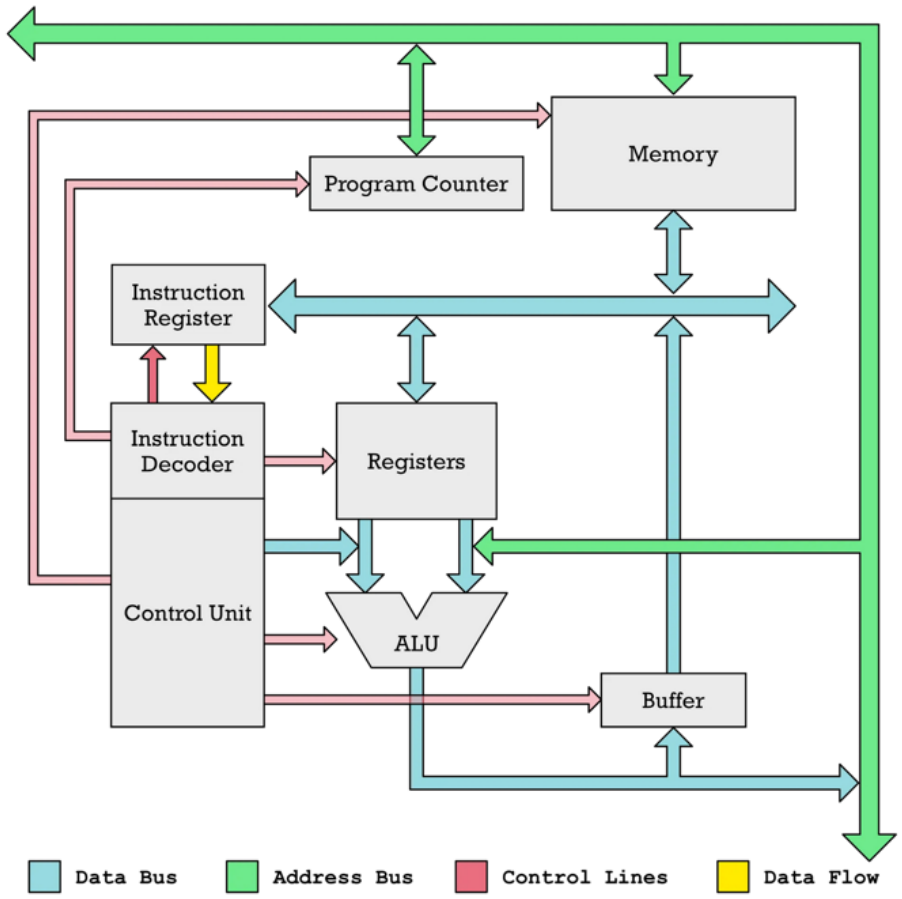
```
False       await       else        import      pass
None        break       except      in          raise
True        class       finally     is          return
and         continue    for         lambda      try
as          def         from        nonlocal    while
assert      del         global      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

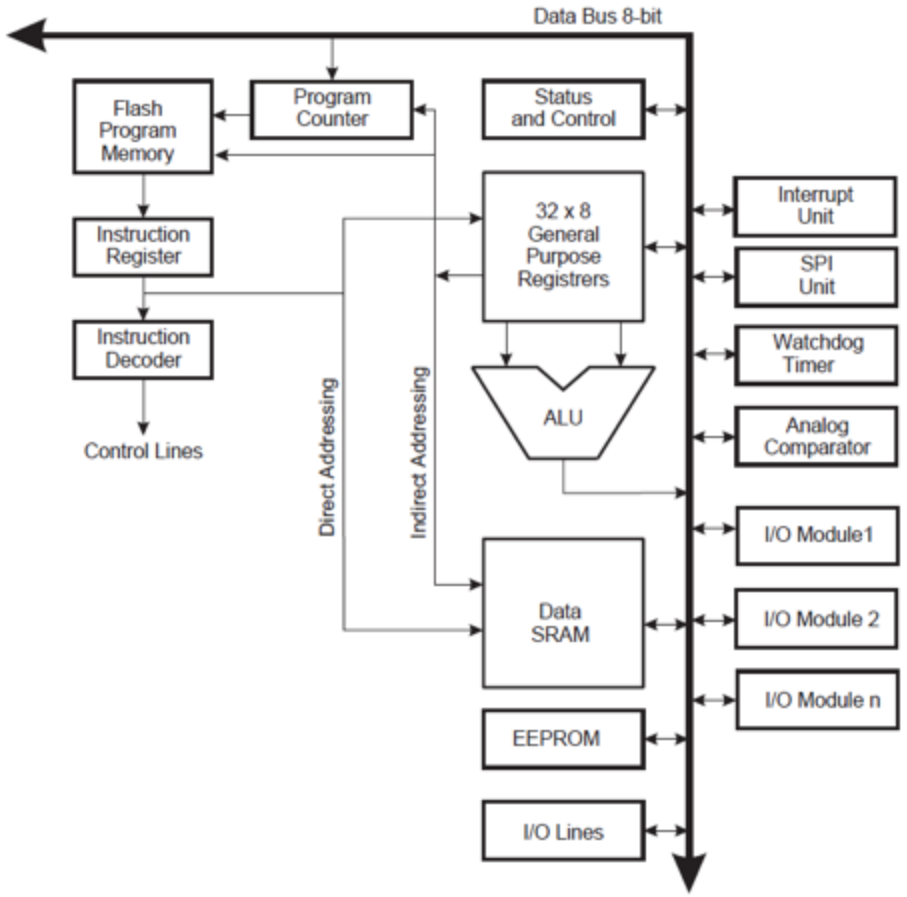
Aufbau und Funktion eines Microprozessors



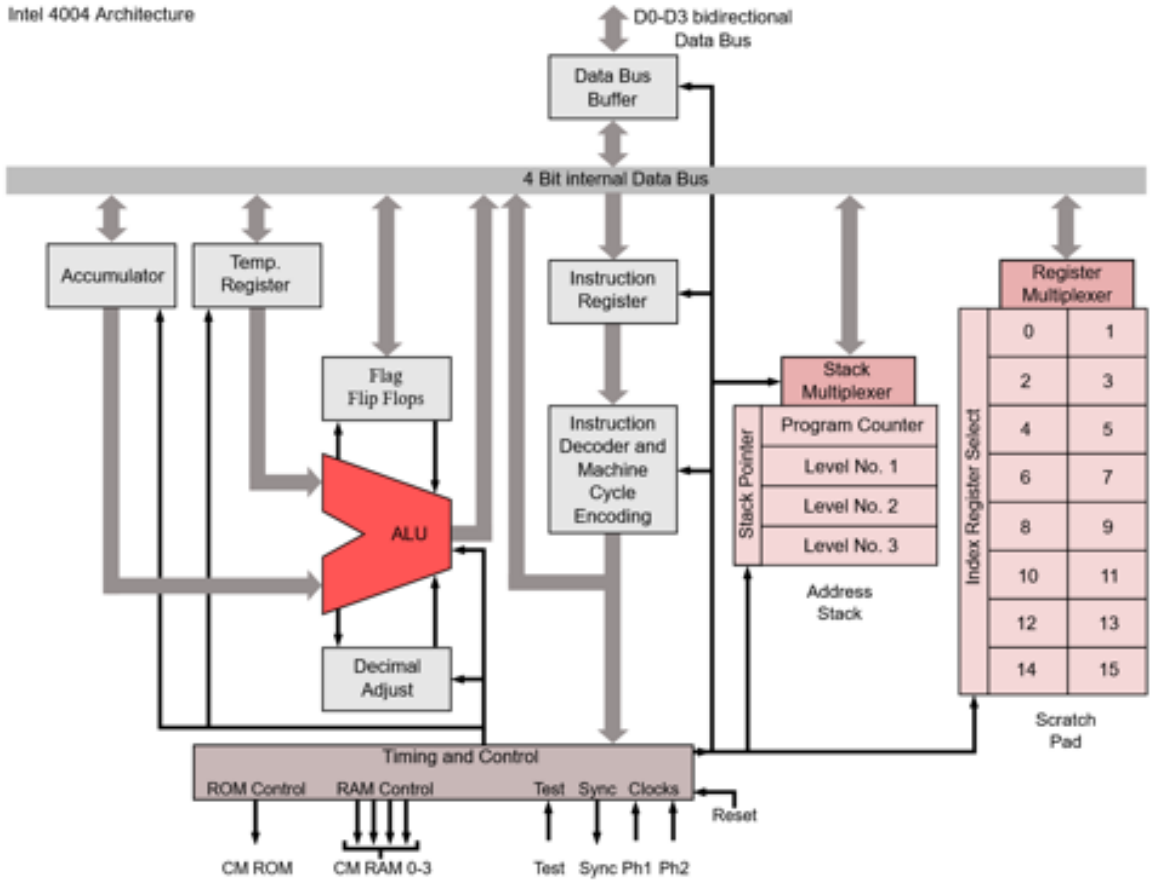
Decoding instruction located in instruction register.

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

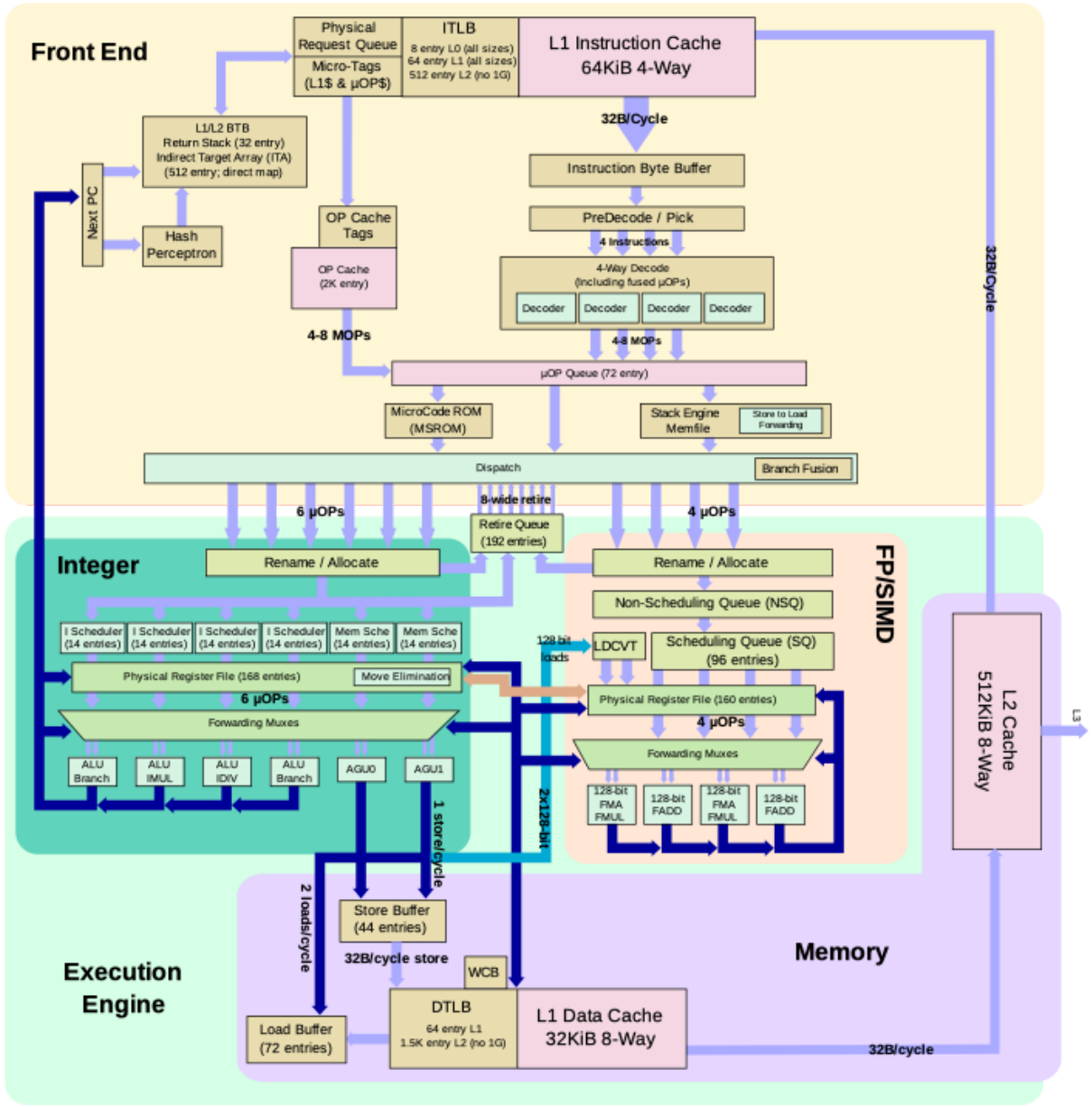
AVR Architektur Blockschaltbild



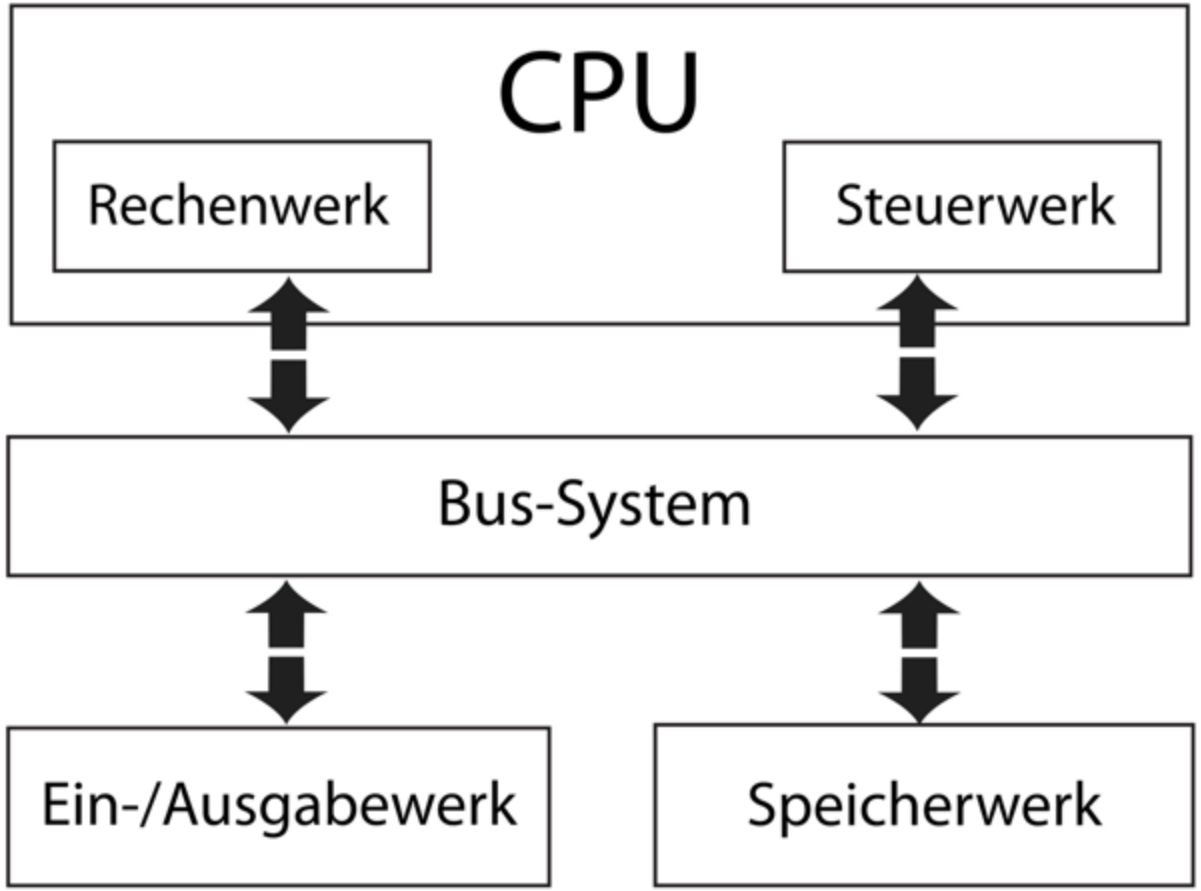
1971: Intel 4004



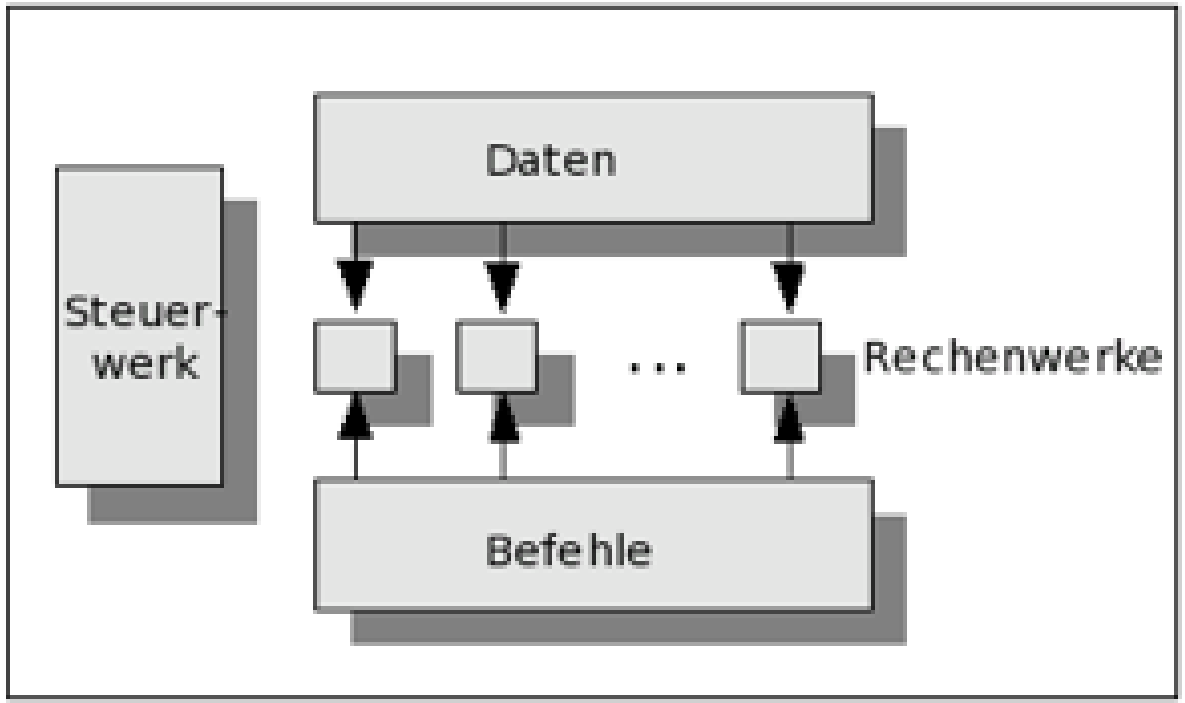
AMD Threadripper



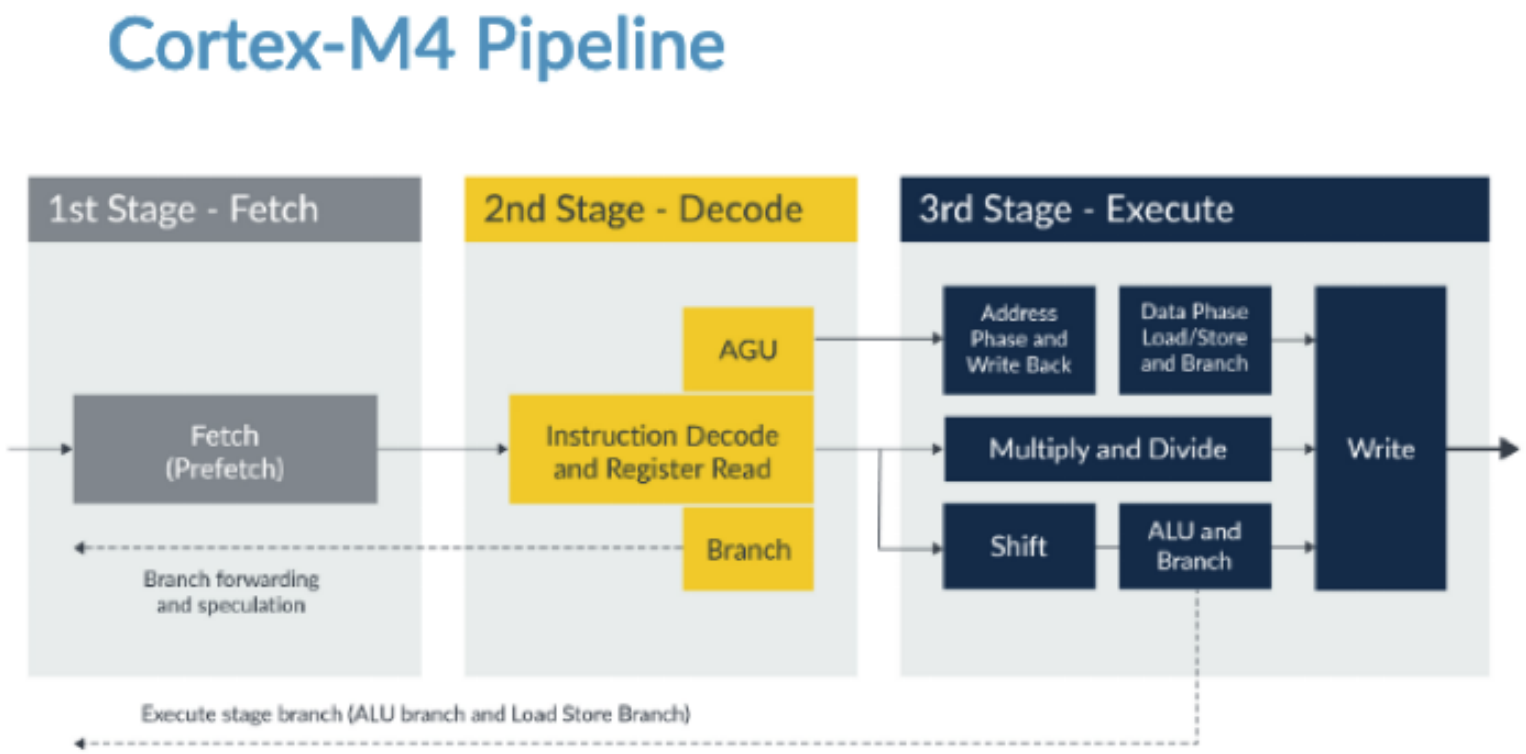
von Neumann Architektur



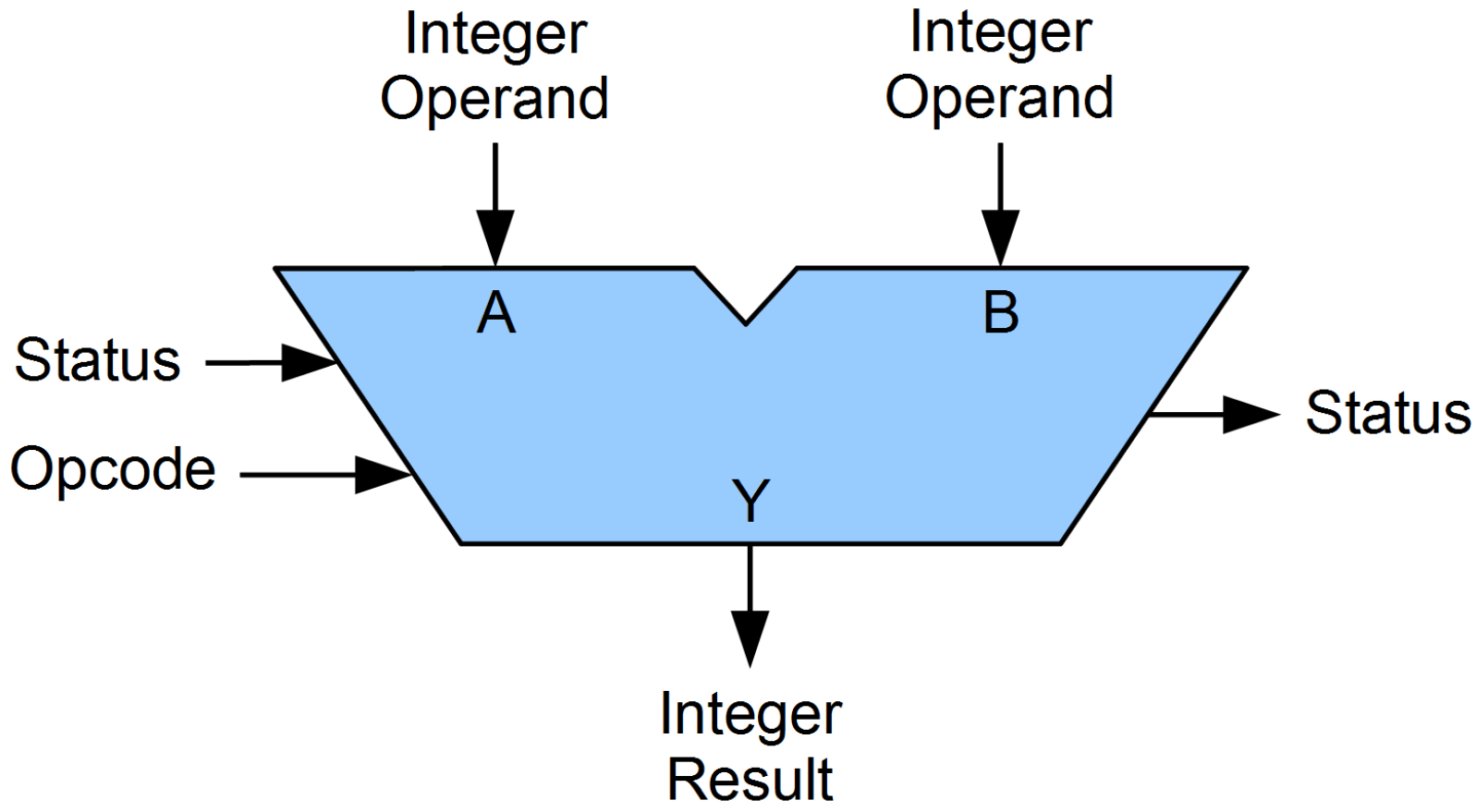
Harvard Architektur



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:

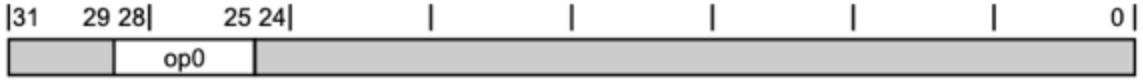
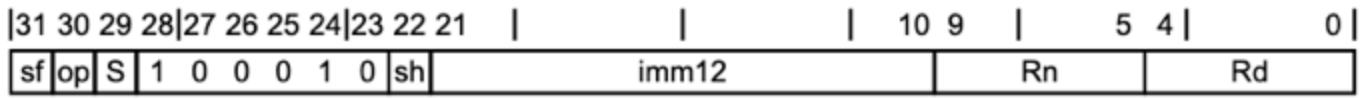


Table C4-1 Main encoding table for the A64 instruction set

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

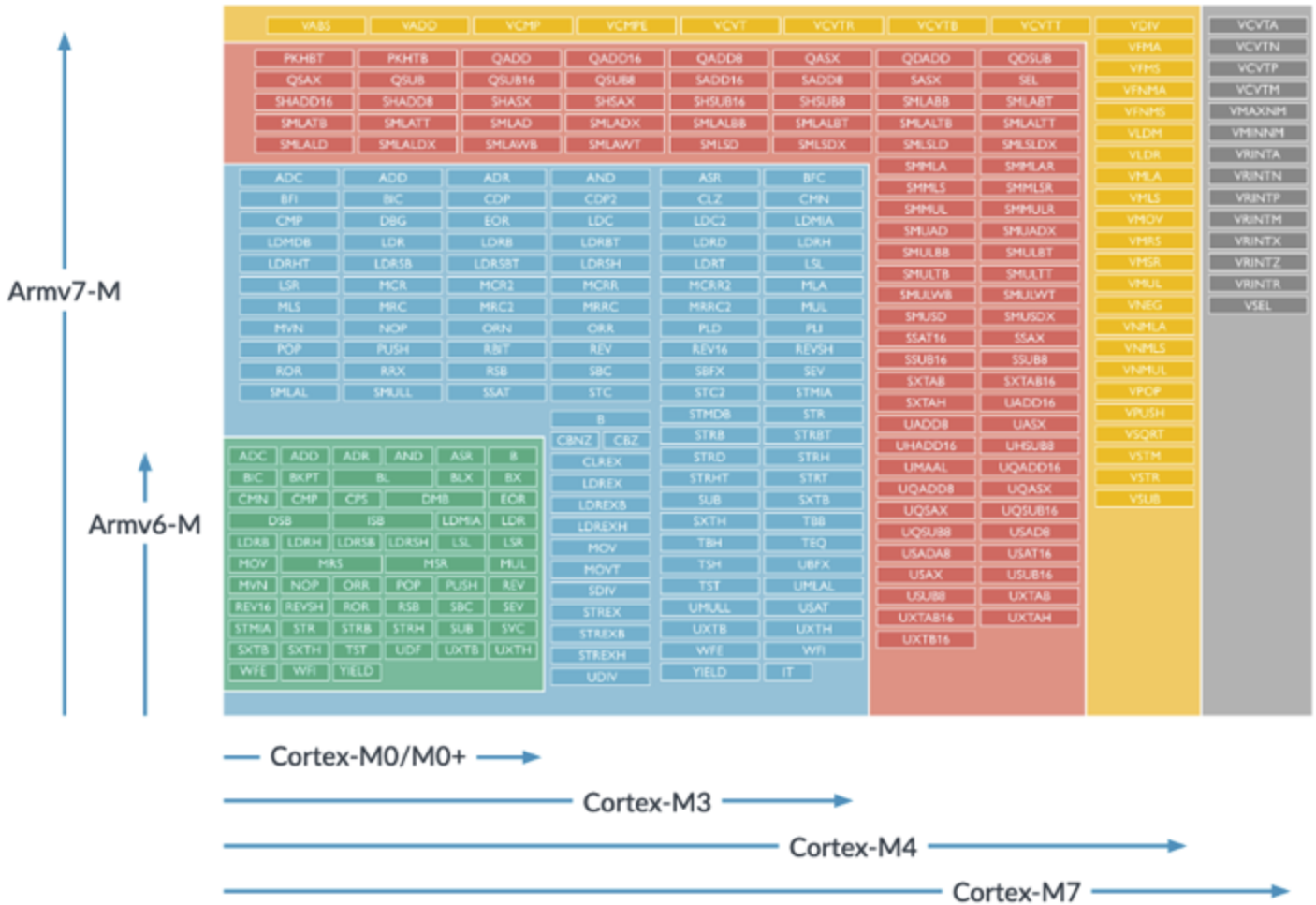


Figure 5: Instruction set

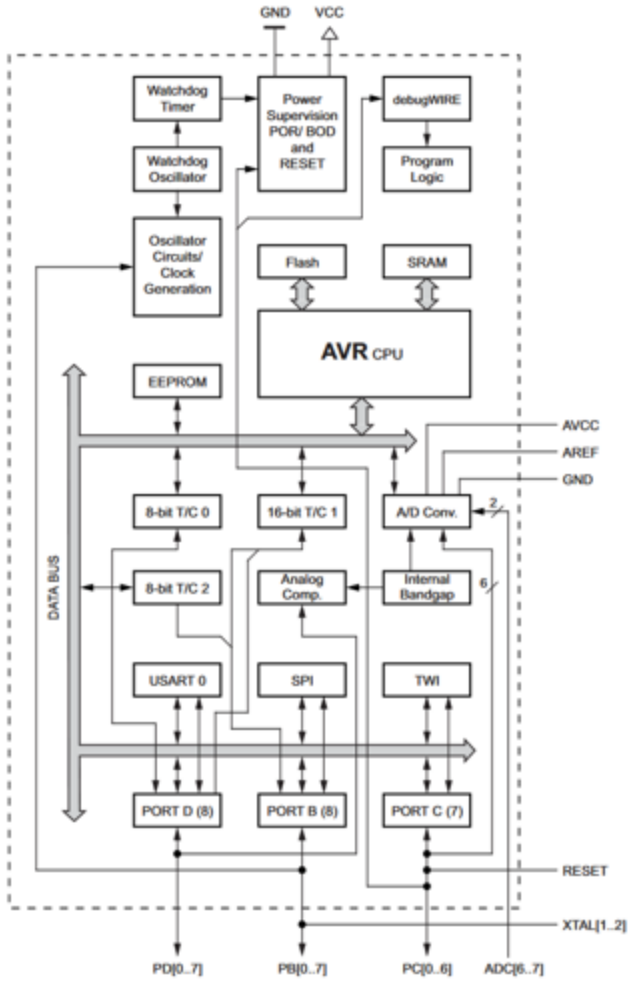
Assembler

```
mov a, 10000 ; Grenzwert der Drehzahl
mov b, 30    ; Grenzwert der Temperatur
mov O,1      ; Abschaltsignal

:loop          ; Markierung im Programmfluss (keine Instruktion, wird vom Assembler für Sprungadressen verwendet)
in d,PORT1   ; einlesen der aktuellen drehzahl-Werte
in t,PORT2   ; einlesen der aktuellen temp-Werte

:drehcheck
cmp d,a      ; prüfe die Drehzahl
jg  tempcheck; wenn Grenzwert nicht erreicht, springe zu :tempcheck
out PORT3,O  ; Grenzwert erreicht! setze Abschaltsignal

:tempcheck
cmp t,b      ; prüfe die Temperatur
jg  loop     ; wenn Grenzwert nicht erreicht, springe zu :loop
out PORT3,O  ; Grenzwert erreicht! setze Abschaltsignal

jmp loop     ;unbedingter Sprung zur Marke :loop (Endlosschleife)
```

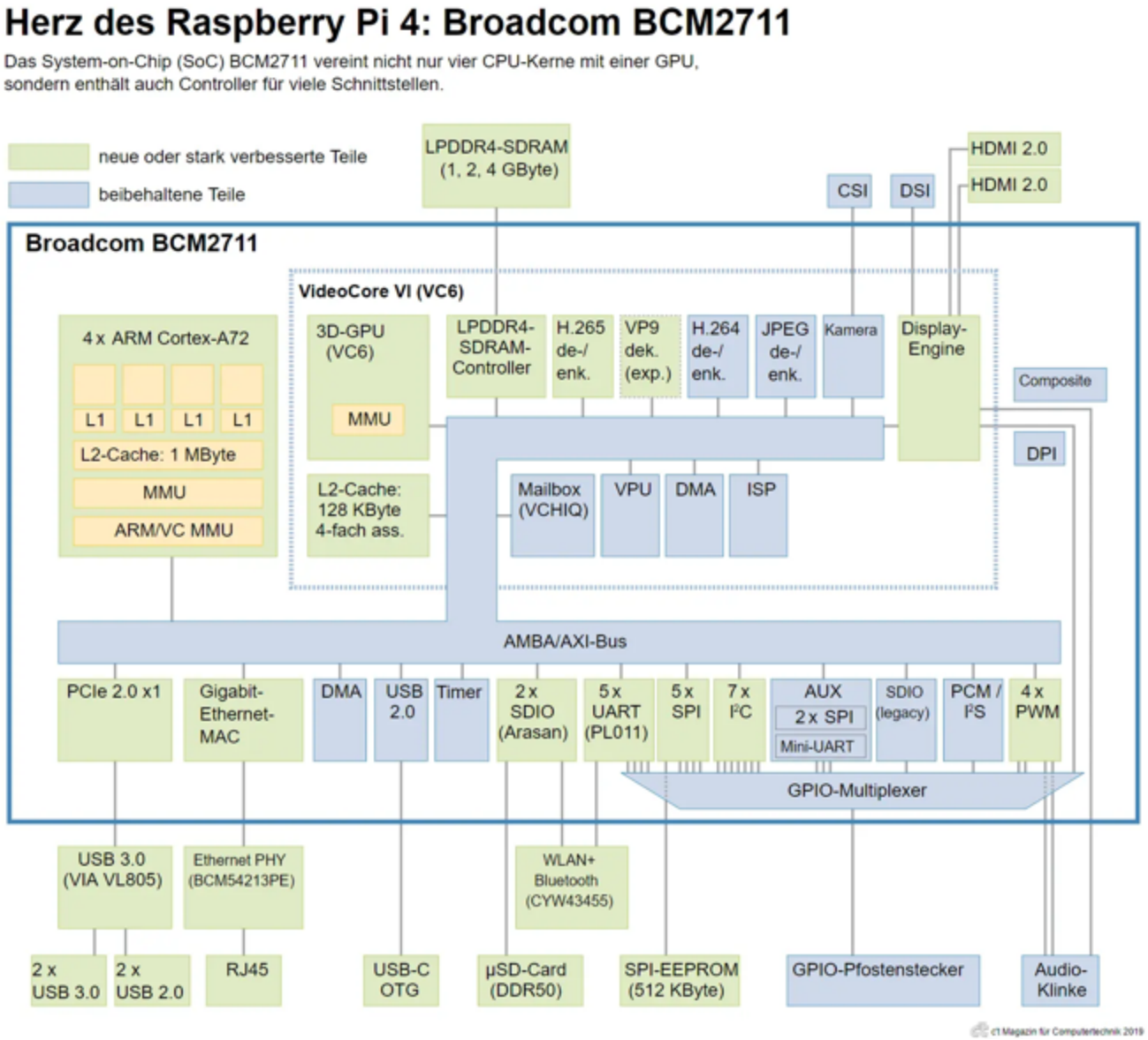
https://de.wikipedia.org/wiki/Echtzeitsystem

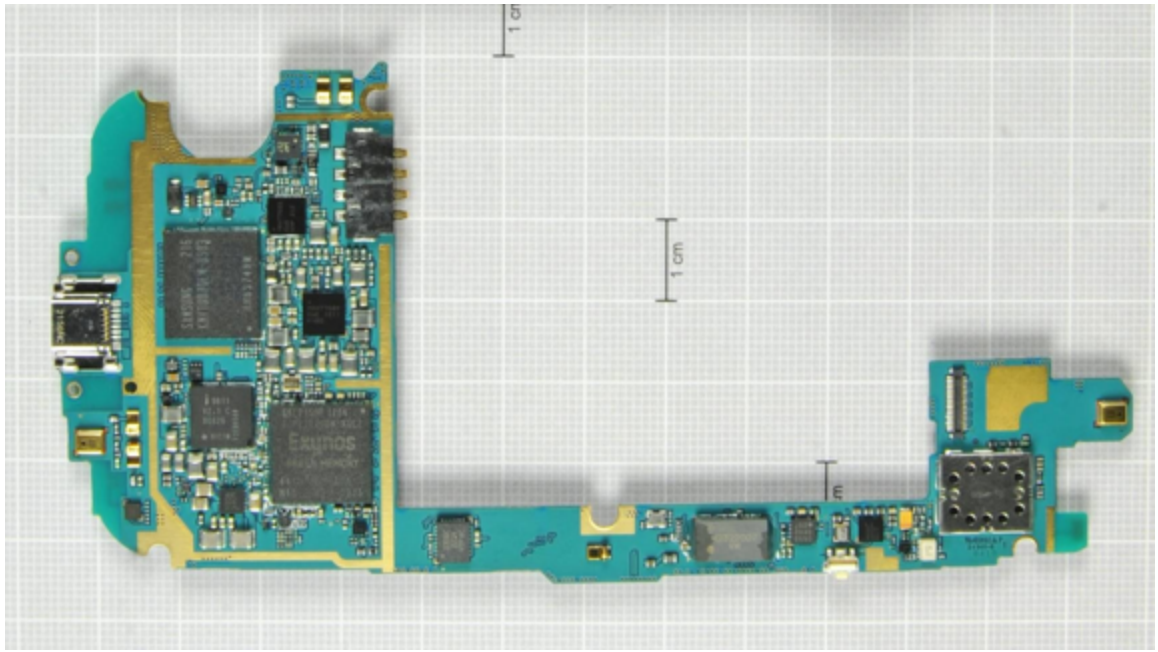
SoC vs Microprocessor vs Microcontroller

Microcontroller: ATmega328P

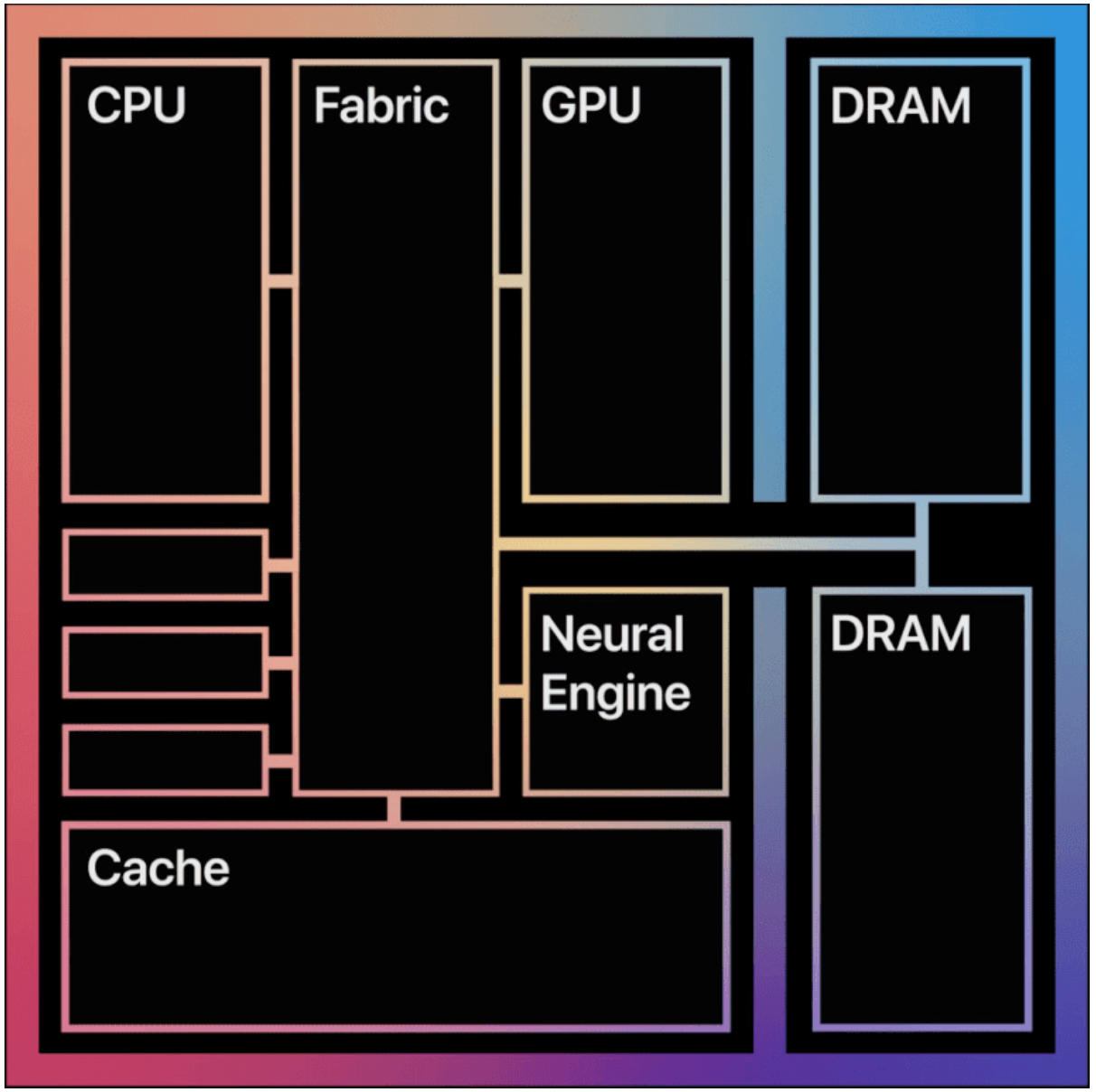


System on Chip (SoC)



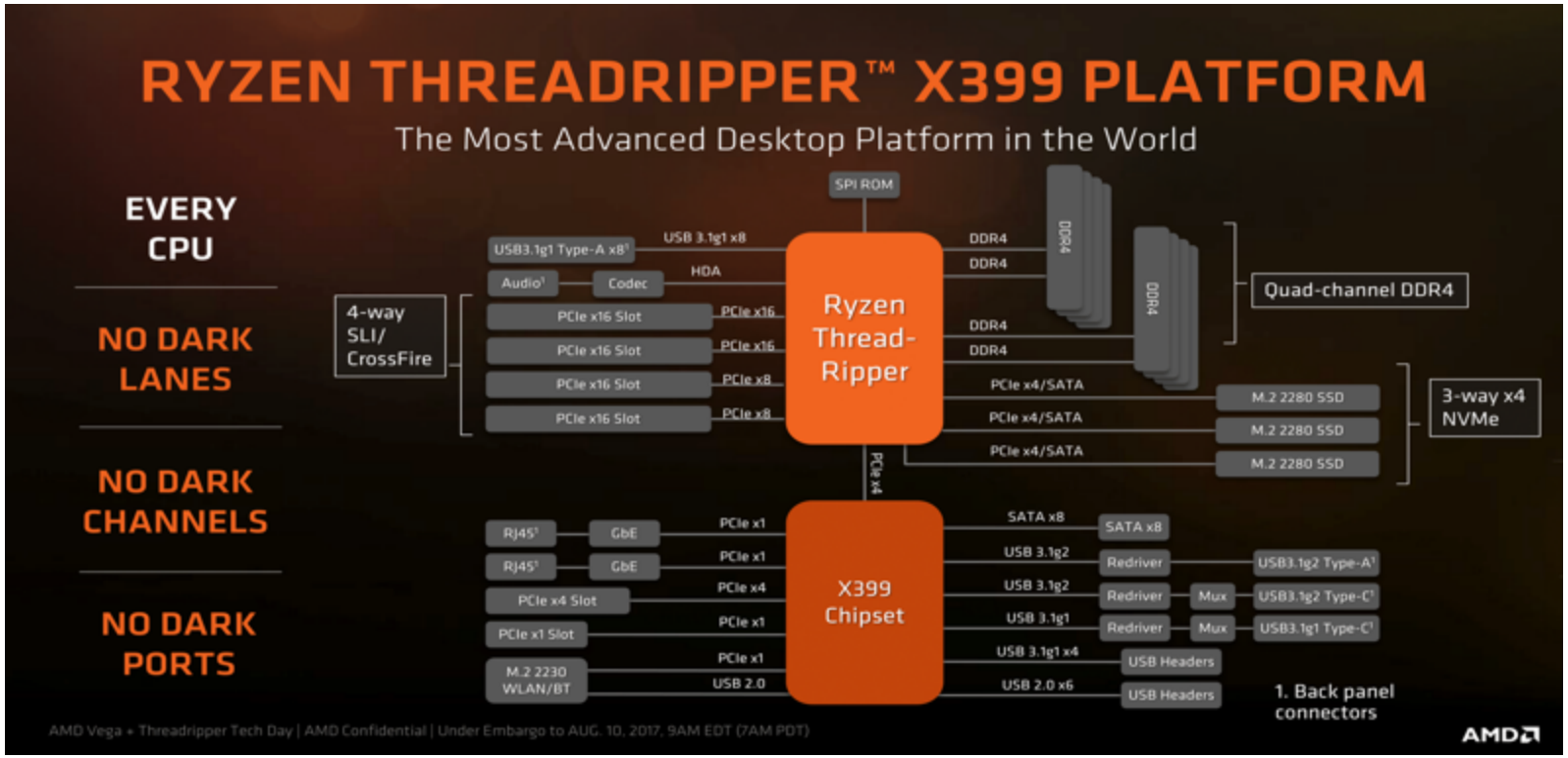


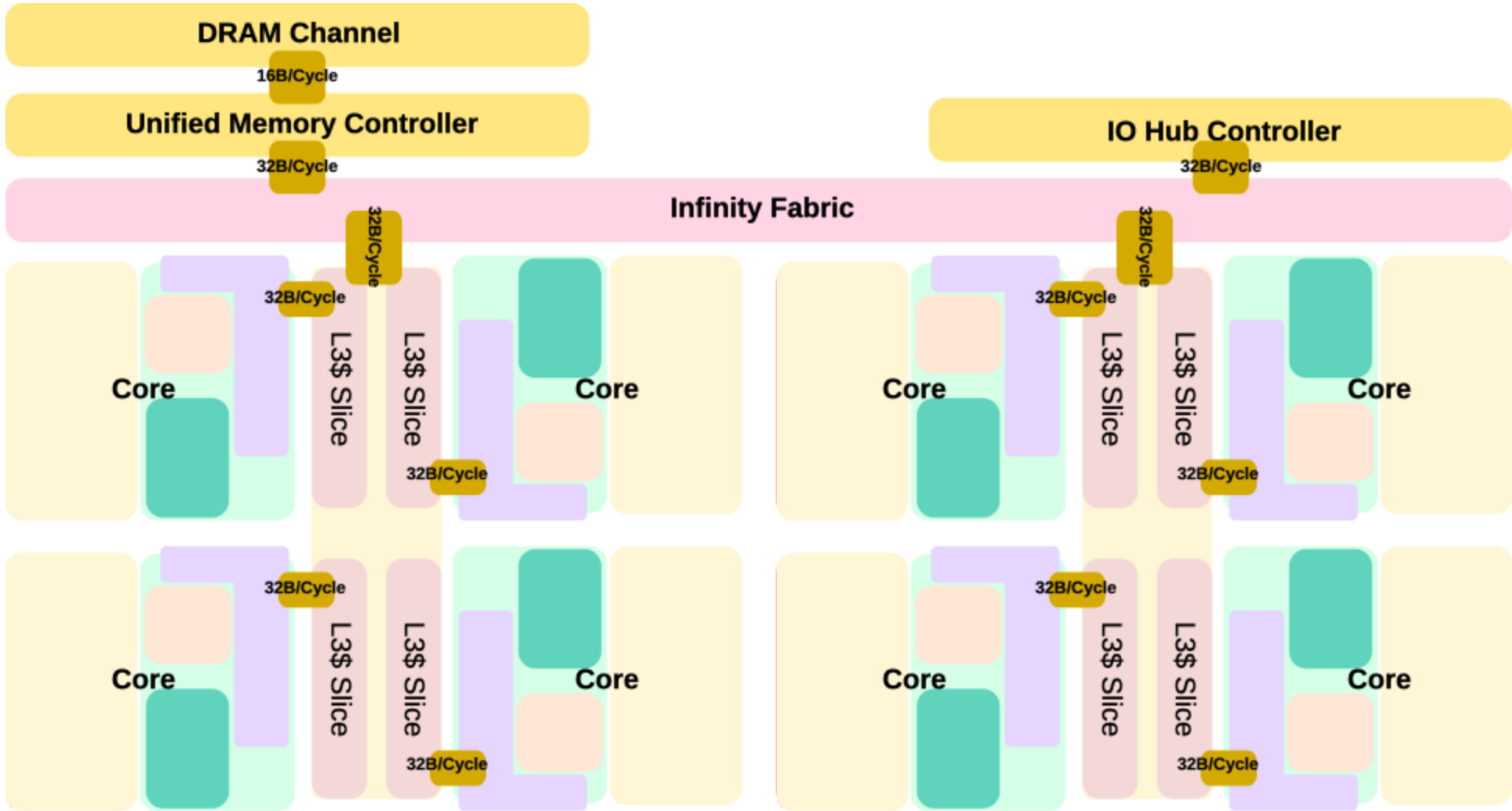
Samsung Galaxy S3



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)

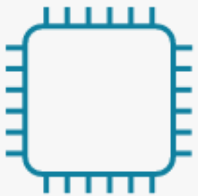
> "Arm licenses processor designs to semiconductor companies that incorporate the technology into their computer chips.
> Licensees pay an up-front fee to gain access to our technology, and a royalty on every chip that uses one of our technology designs.
> Typically, the royalty is based on the selling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

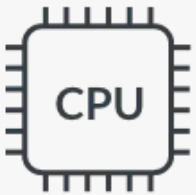


270Bn+

Arm-based chips shipped to date

99%

of smartphones run on Arm-based processors



50%

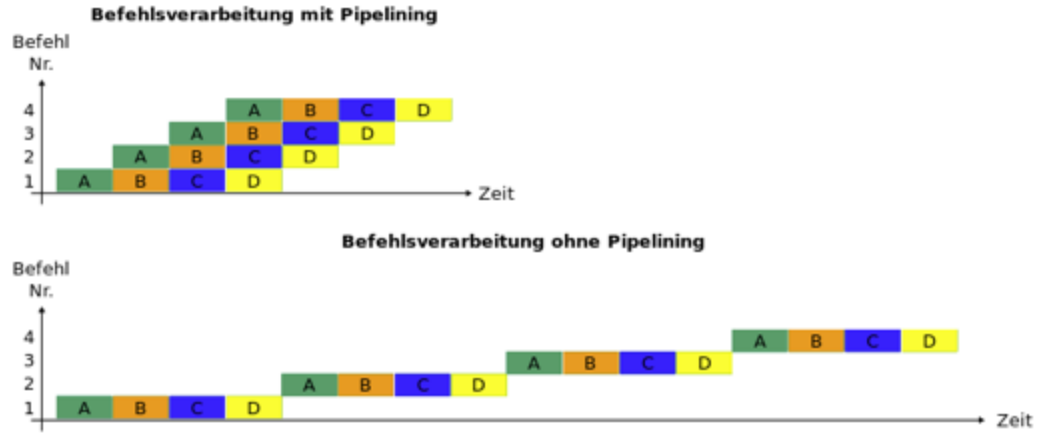
of all chips with processors are Arm-based

RISC vs CISC

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

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)

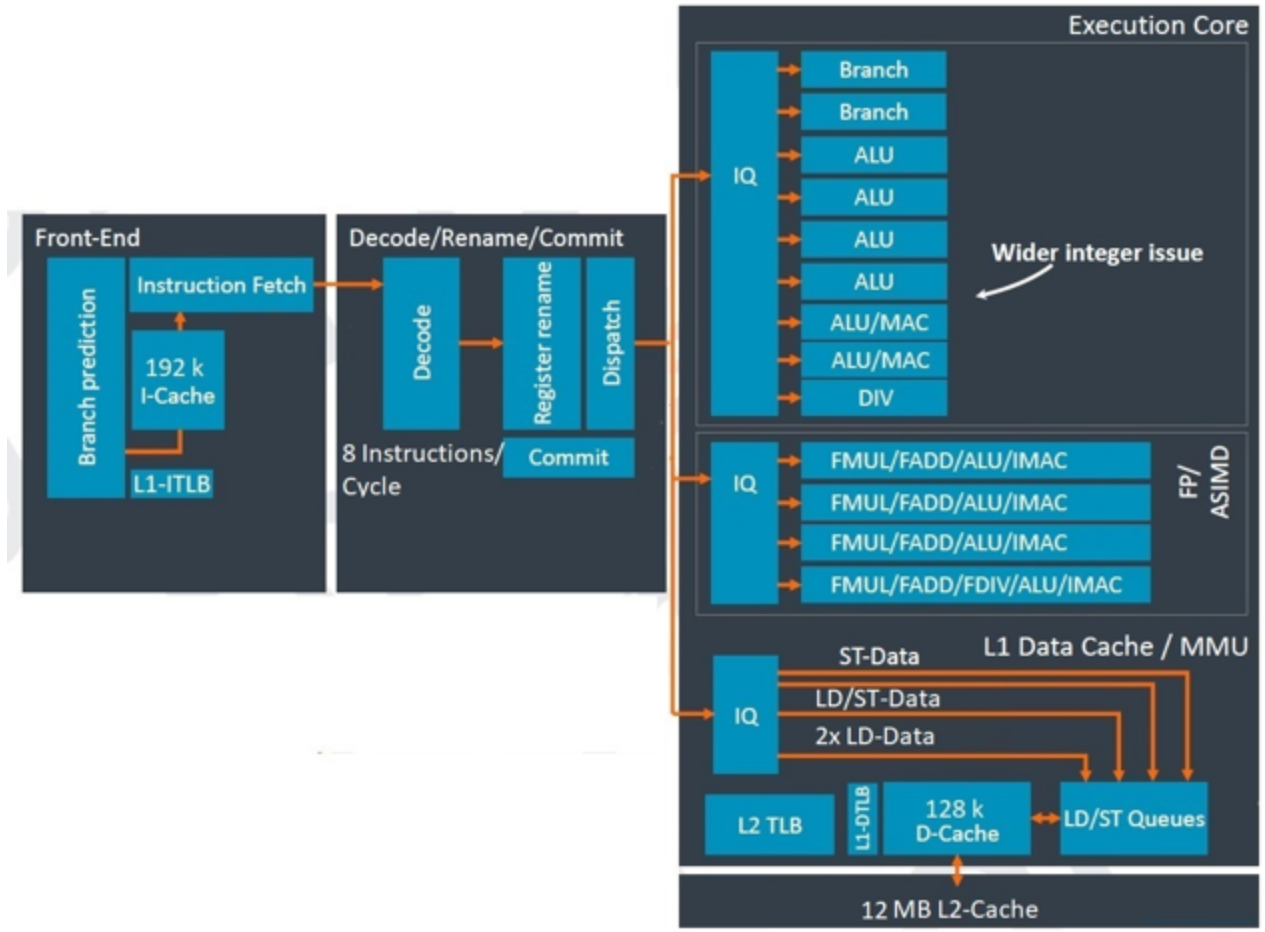
In der Dekodier- und Ladephase wird der geladene Befehl dekodiert (1. Takthälfte) und die notwendigen Daten aus dem Arbeitsspeicher und dem Registersatz geladen (2. Takthälfte).

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.


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