

# **Microcomputertechnik**

# Überblick



# Alan Touring

- \*1912 - †1954
- Begründer der «Computer Science»
- WWII: Massgeblich am Knacken der deutschen Enigma-Verschlüsselung beteiligt
- Verfolgt und «therapiert» wegen Homosexualität
- Vermutlich Selbstmord
- The Imitation Game, Benedict Cumberbatch

# Turingmaschine

Die Turingmaschine hat ein Steuerwerk, in dem sich das Programm befindet, und besteht außerdem aus

- einem unendlich langen Speicherband mit unendlich vielen sequentiell angeordneten Feldern.
- einem programmgesteuerten Lese- und Schreibkopf, der sich auf dem Speicherband feldweise bewegen und die Zeichen verändern kann.
- Turing bewies, dass solch ein Gerät in der Lage ist, „jedes vorstellbare mathematische Problem zu lösen, sofern dieses auch durch einen Algorithmus gelöst werden kann“.

# Turingvollständigkeit

«Exakt ausgedrückt bezeichnet Turing-Vollständigkeit in der Berechenbarkeitstheorie die Eigenschaft einer Programmiersprache oder eines anderen logischen Systems, sämtliche Funktionen berechnen zu können, die eine universelle Turingmaschine berechnen kann.»  
[\(https://de.wikipedia.org/wiki/Turing-Vollständigkeit\)](https://de.wikipedia.org/wiki/Turing-Vollständigkeit)

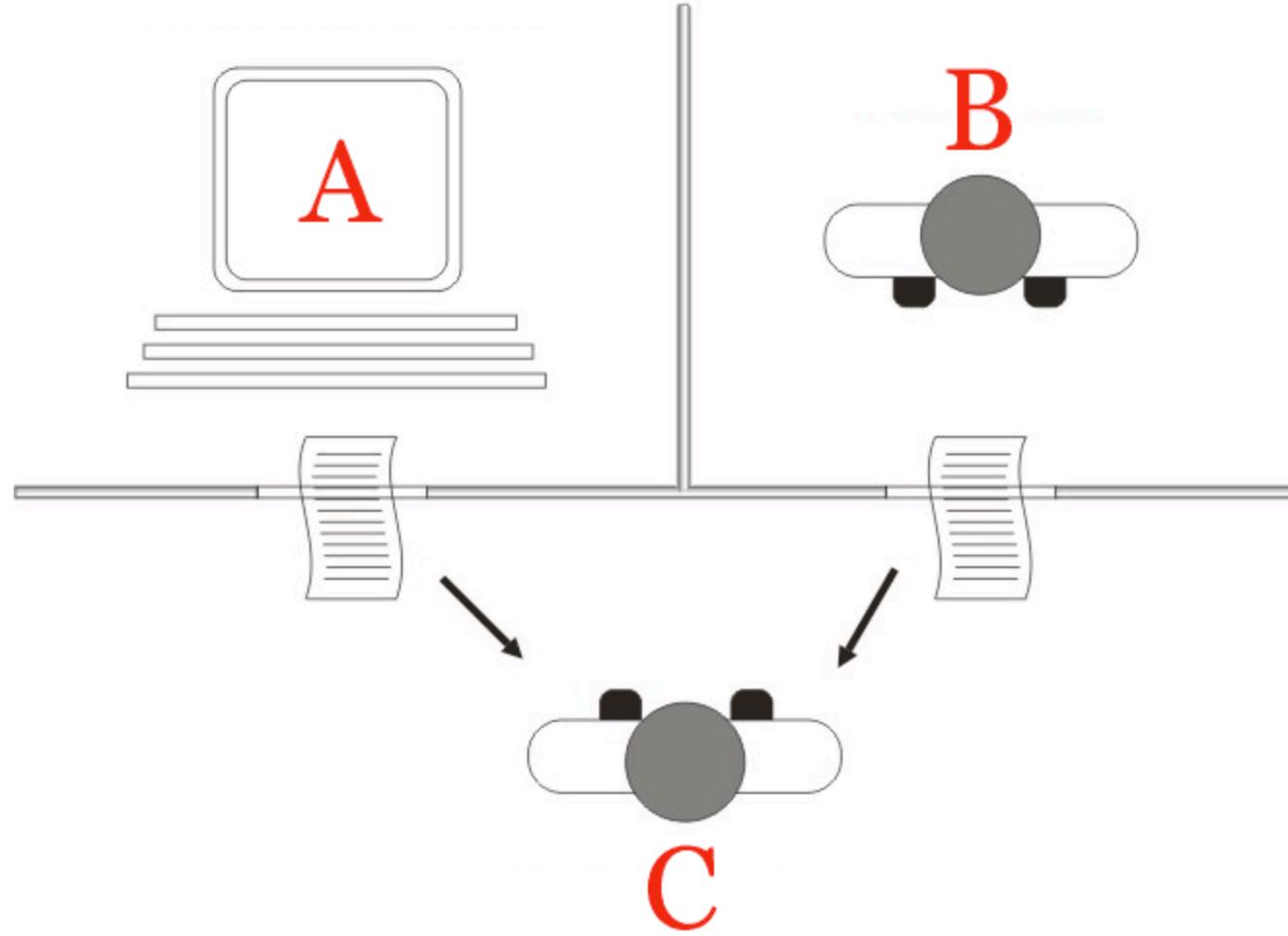
## Halteproblem

«Das Halteproblem beschreibt die Frage, ob die Ausführung eines Algorithmus zu einem Ende gelangt. Obwohl das für viele Algorithmen leicht beantwortet werden kann, konnte der Mathematiker Alan Turing beweisen, dass es keinen Algorithmus gibt, der diese Frage für alle möglichen Algorithmen und beliebige Eingaben beantwortet.»

(<https://de.wikipedia.org/wiki/Halteproblem>)

Wir müssen sicherstellen, dass unsere Programme nicht unabsichtlich endlos weiterlaufen!

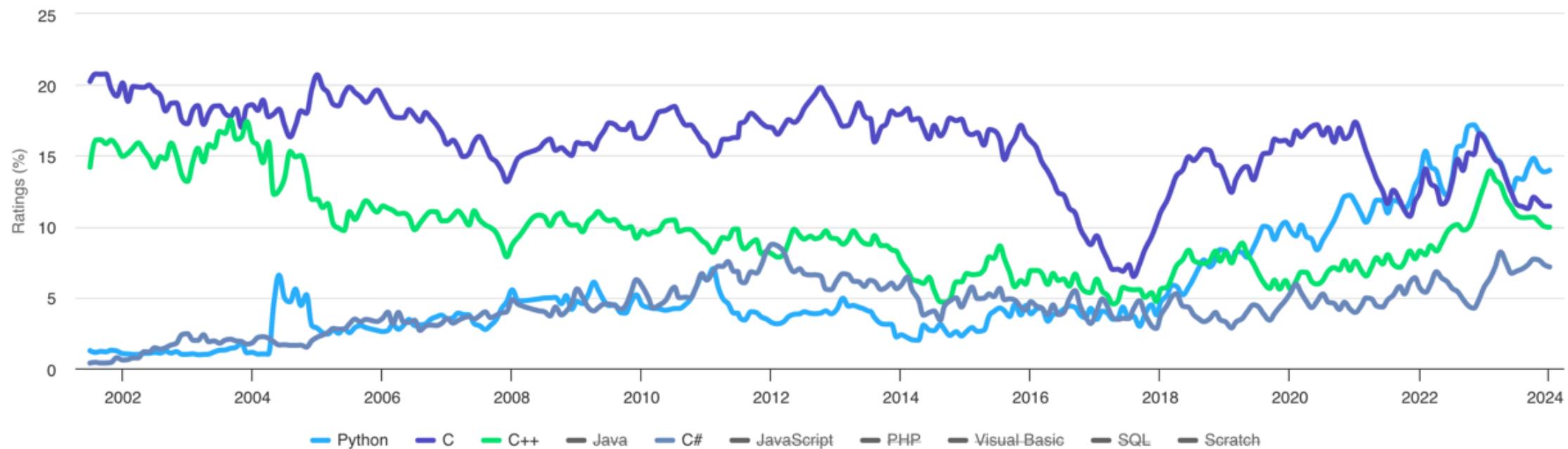
# Turing-Test

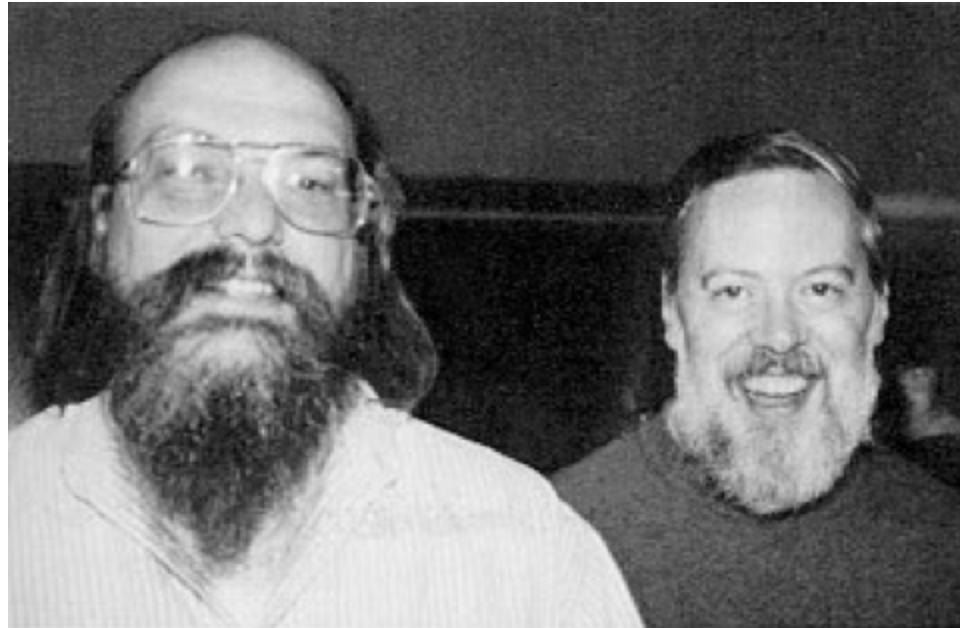


# Software

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)





Ken Thompson, Dennis Ritchie

## C Keywords (Auswahl)

bool (C23)    extern  
false (C23)    float  
break  
case  
char  
const  
continue

for  
goto  
if  
int  
long

sizeof  
static  
struct  
switch  
true (C23)  
typedef

default  
do  
double  
else  
unsigned  
void

return  
volatile  
short  
signed  
register  
union

# 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

## 4. Generation

- SQL
- Unix Shell
- LabVIEW
- Stata
- R
- MATLAB
- MaxMSP

# Programmiersprachen der 3. Generation

- ALGOL
- Cobol
- Fortran
- C, C++
- C#
- Java
- Python
- Go
- Rust
- JavaScript
- etc.

# Rust

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

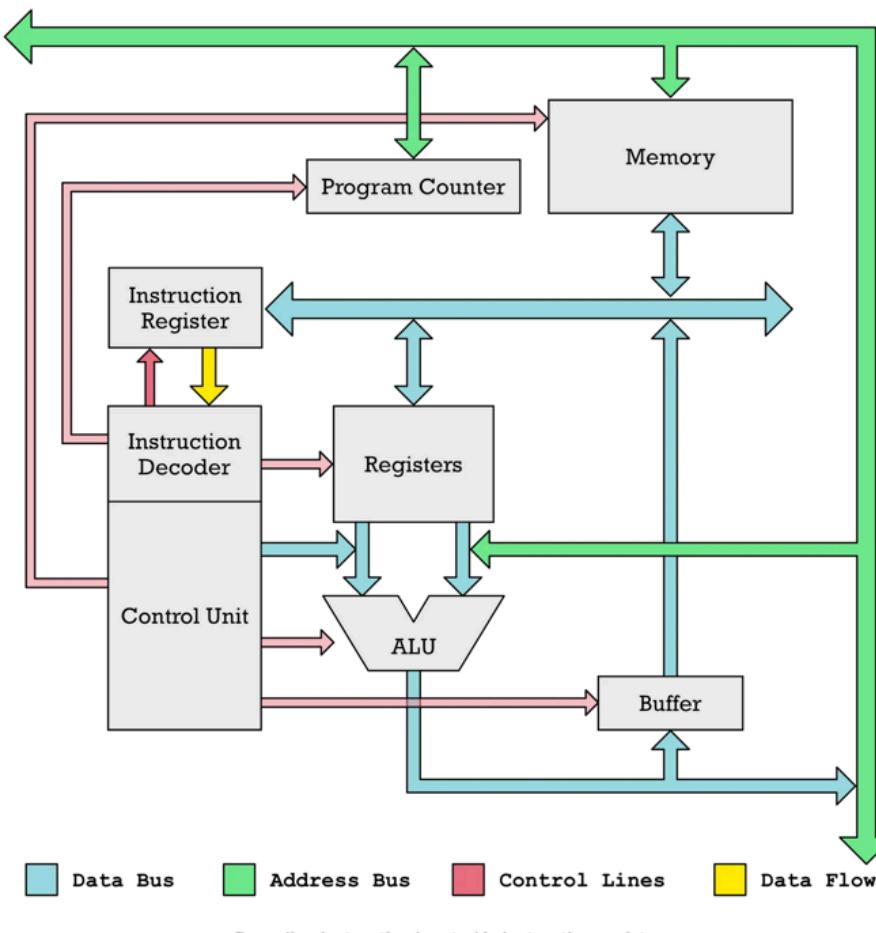
# **Assembler (2. Generation)**

# 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\005\000\000"
str.0:
    .ascii  "attempt to multiply with overflow"
```

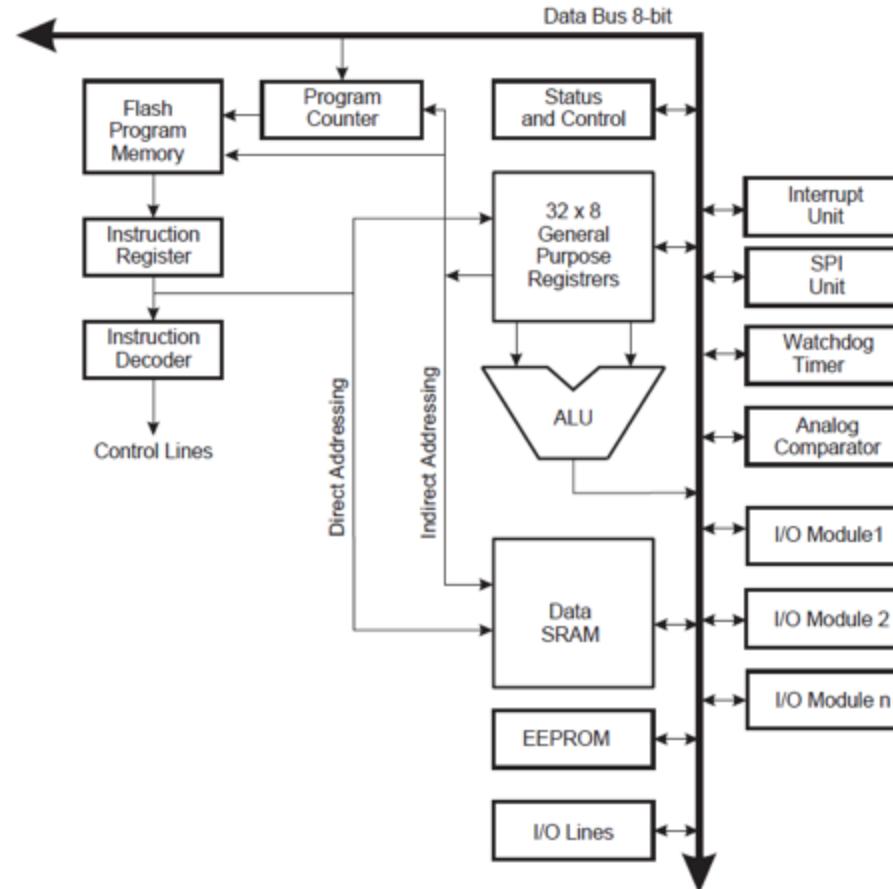
# **Maschinensprache (1. Generation)**

# Aufbau und Funktion eines Microprozessors

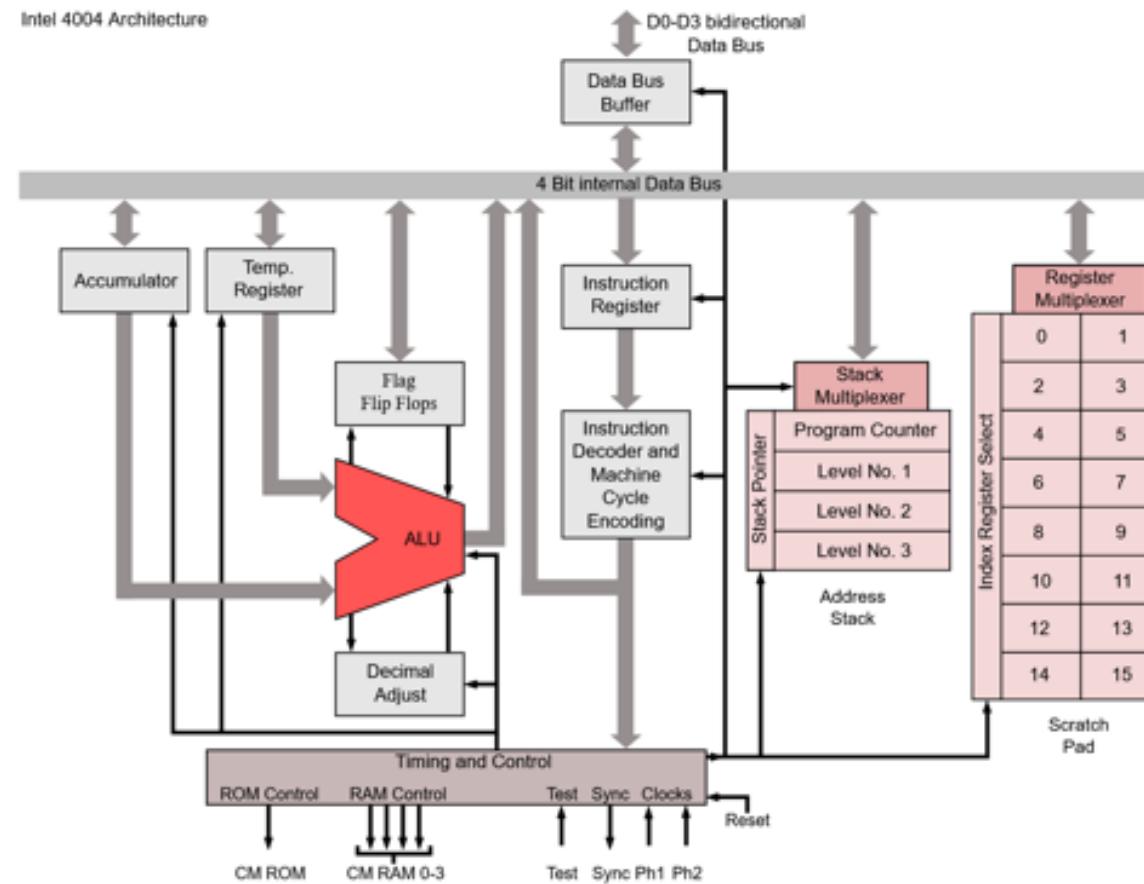


<https://erik-engheim.medium.com/how-does-a-microprocessor-run-a-program-11744ab47d04>

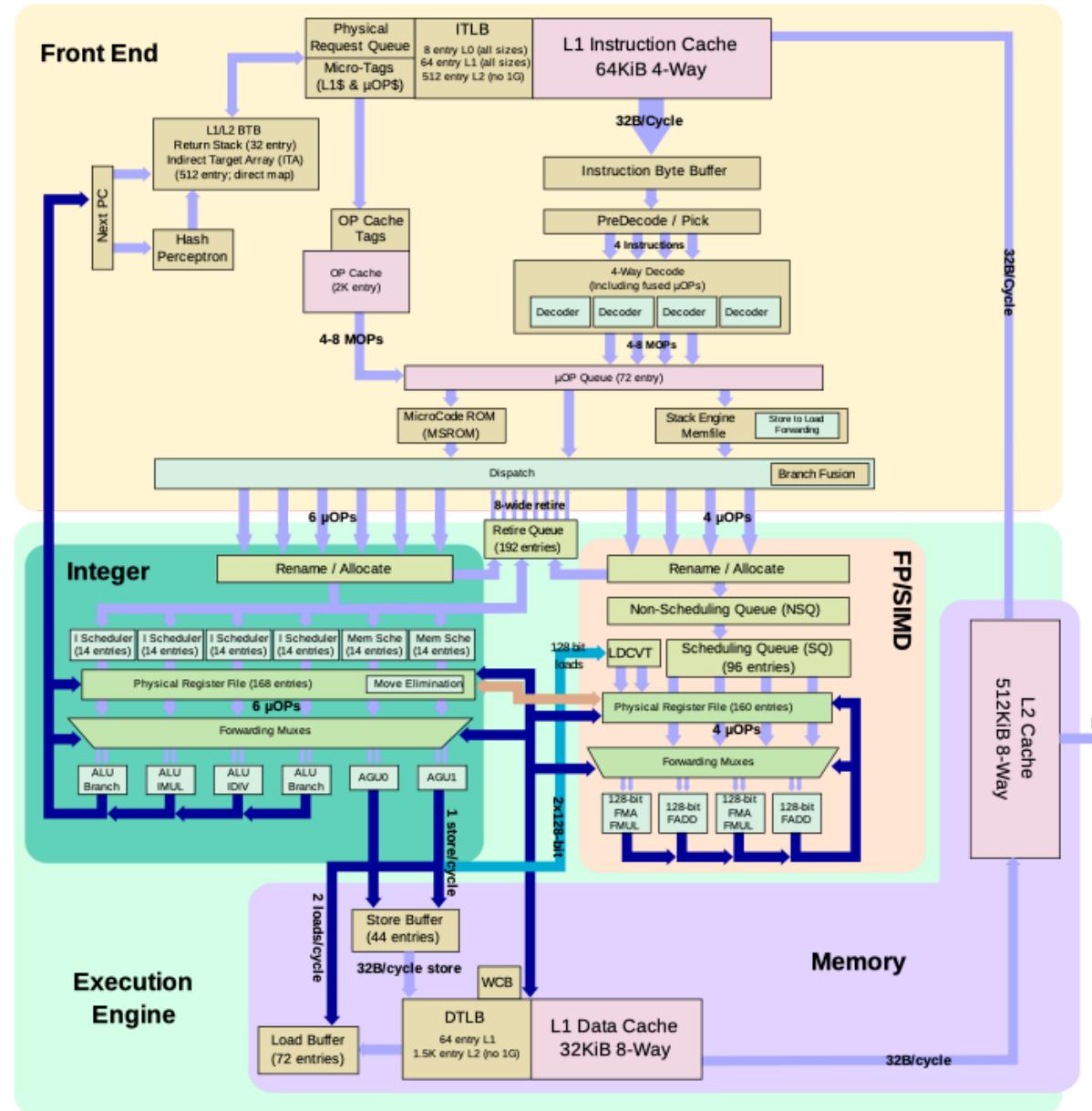
# AVR Architektur Blockschaltbild



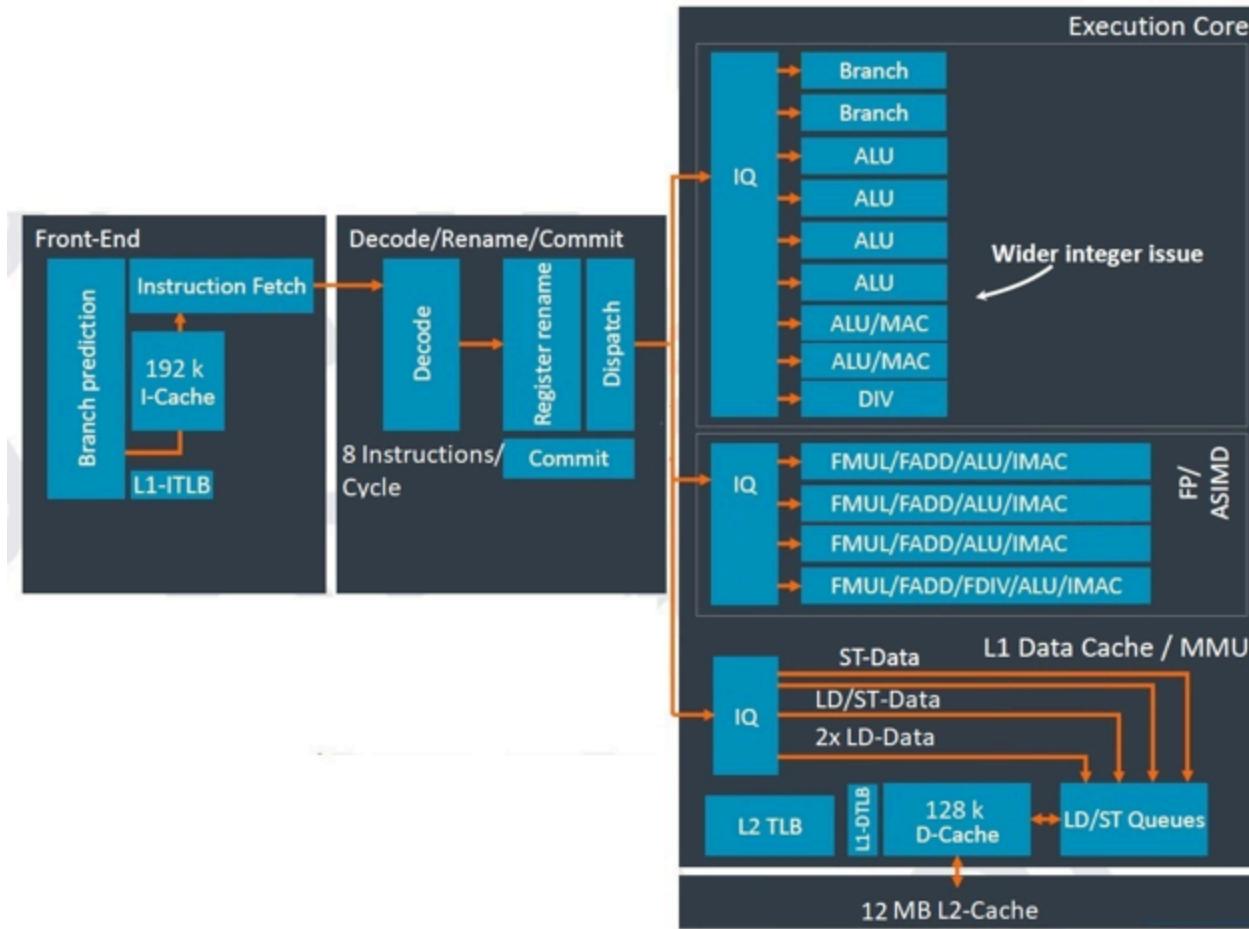
# 1971: Intel 4004



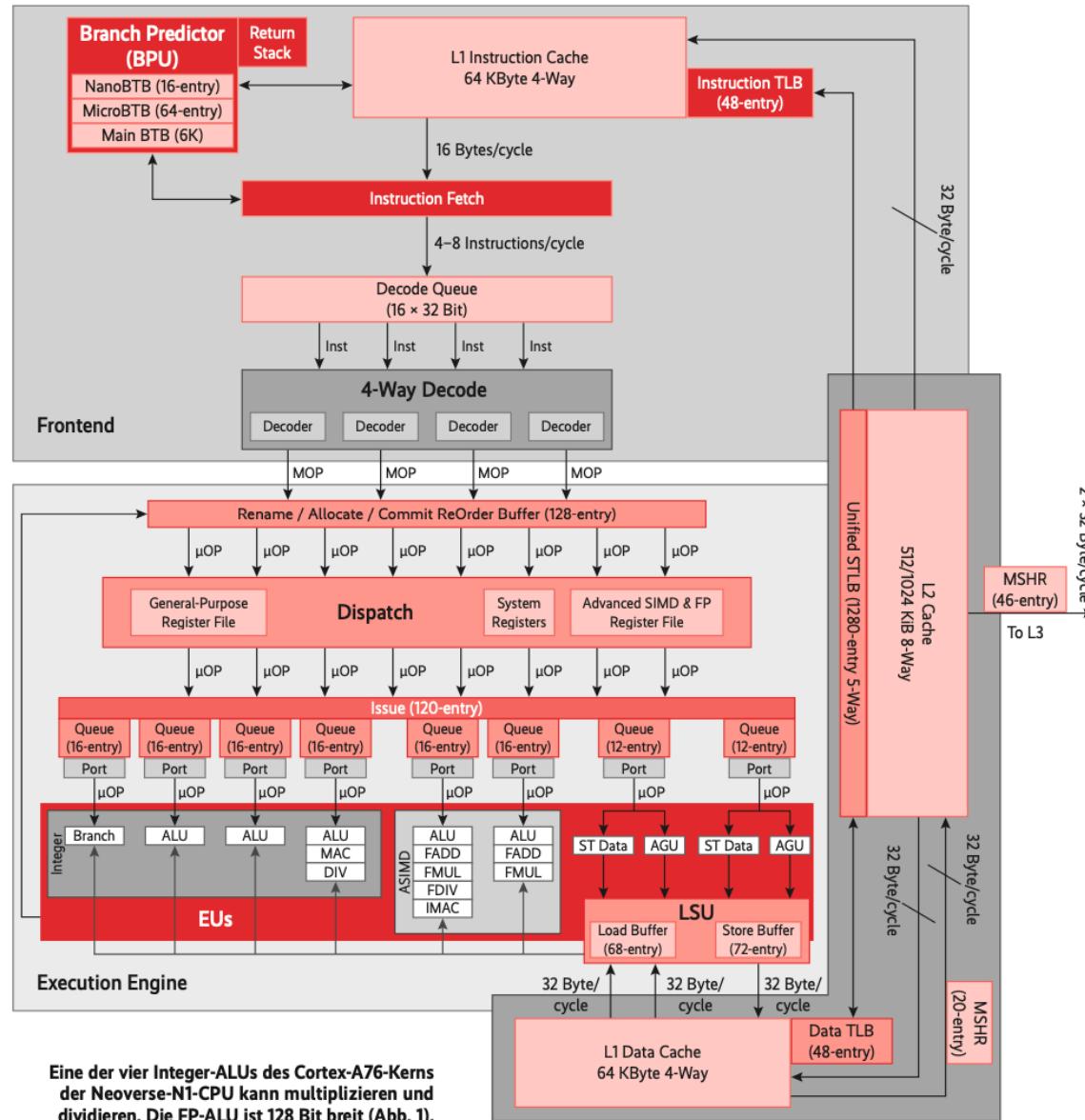
# AMD Threadripper



# Apple M1

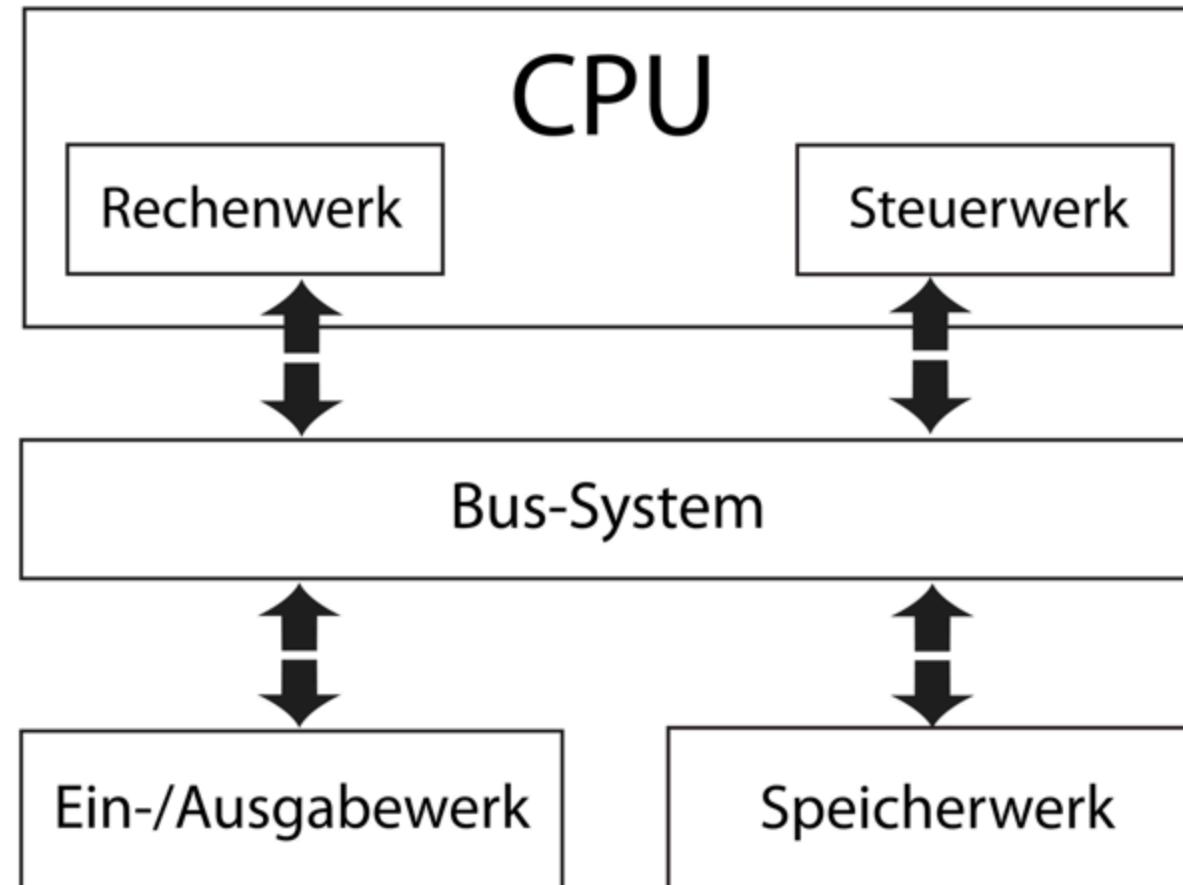


# ARM Cortex A67

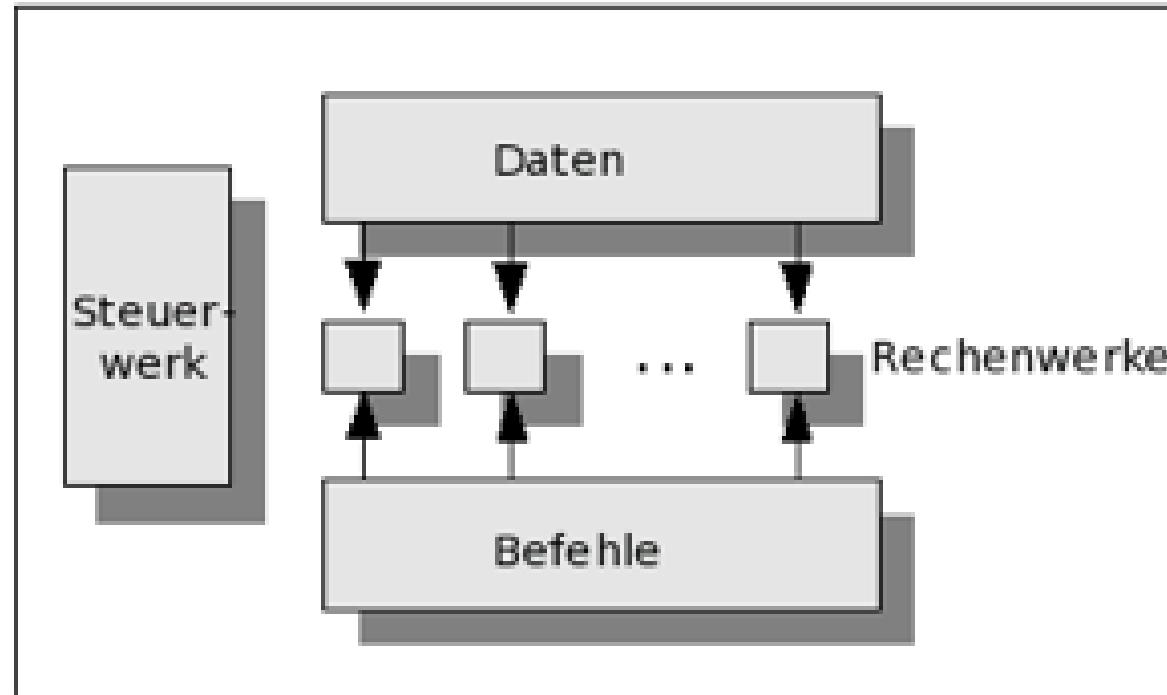


# von Neumann Architektur

- Befehle werden aus einer Zelle des Speichers gelesen und dann ausgeführt.
- Normalerweise wird dann der Inhalt des Befehlszählers um Eins erhöht.
- Es gibt Verzweigungs-Befehle, die in Abhängigkeit vom Wert eines Entscheidungs-Bit den Befehlszähler um Eins erhöhen oder um einen anderen Wert verändern

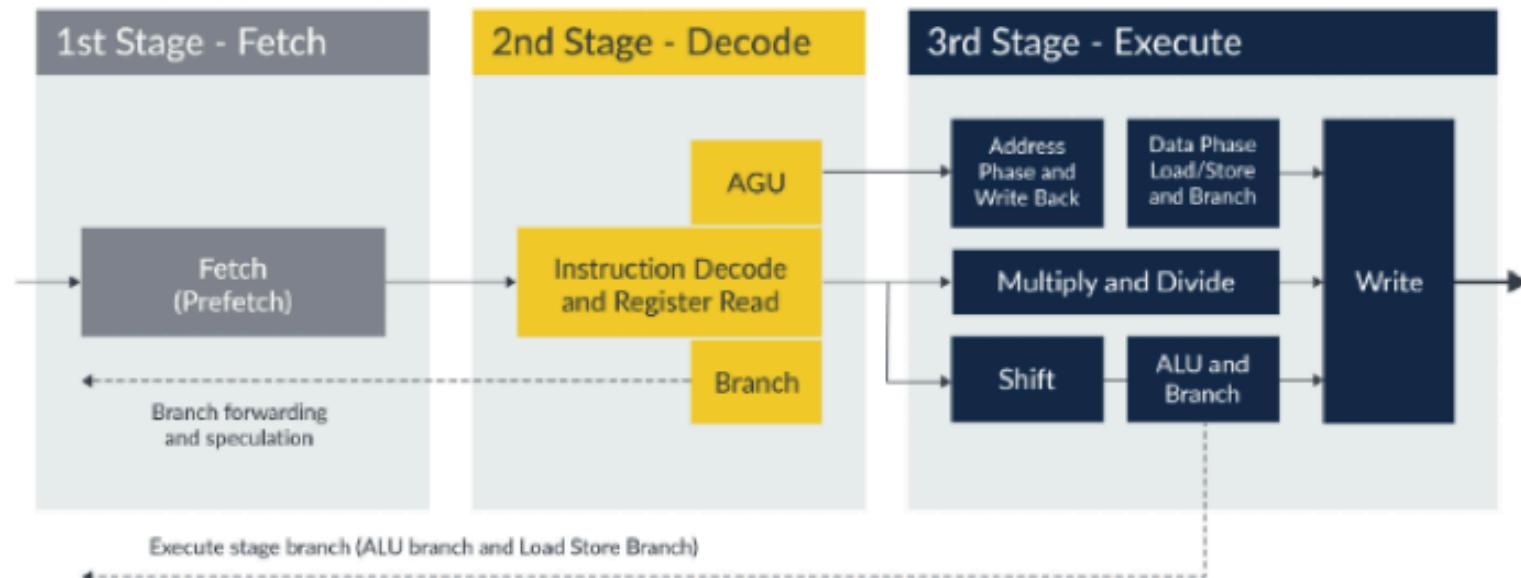


# Harvard Architektur

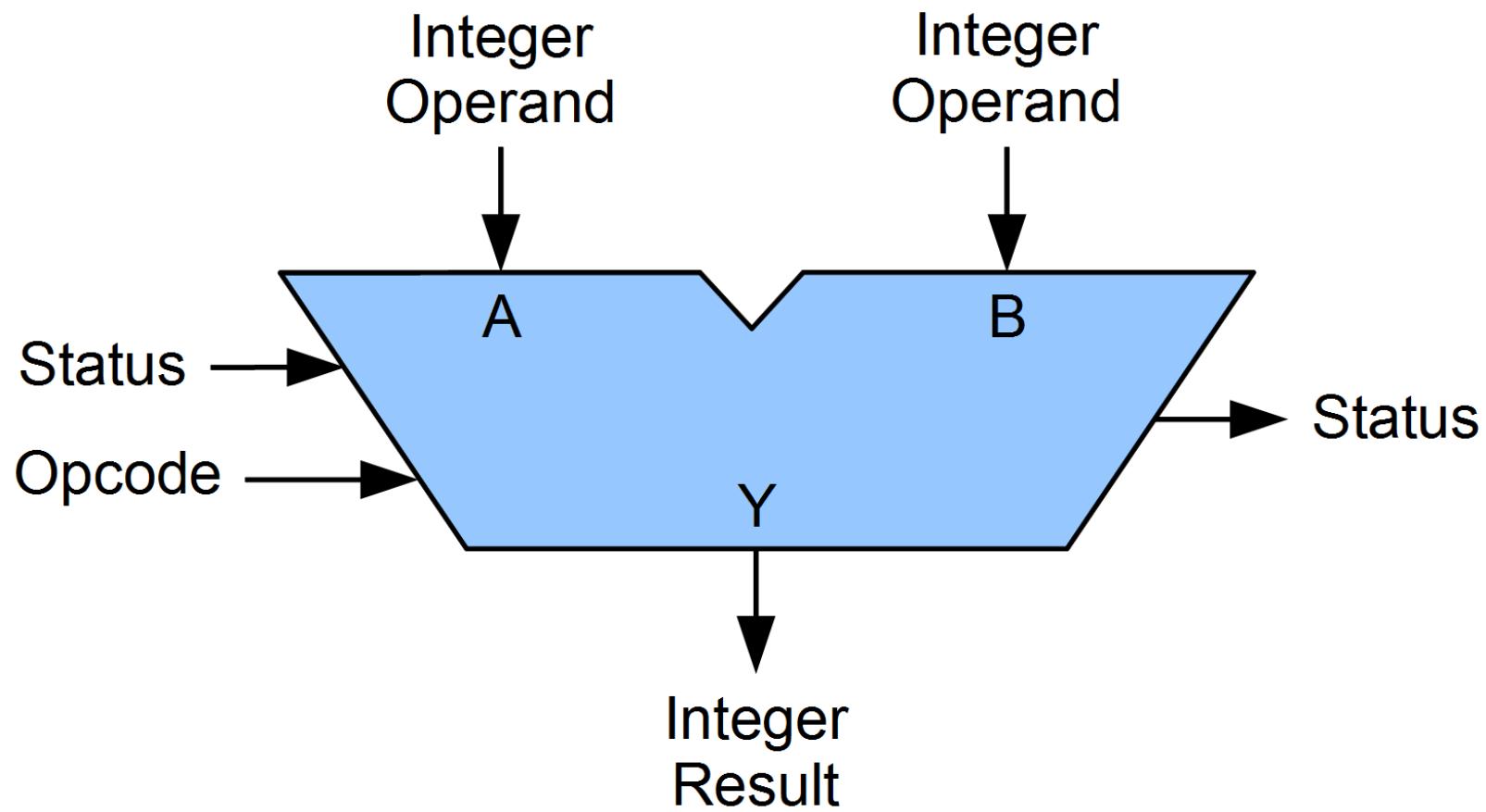


# Fetch - Decode - Execute

## Cortex-M4 Pipeline



## Arithmetic Logic Unit (ALU)



Mindestens:

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

Zusätzlich (Auswahl):

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

# Instruction Set

MIPS32 Add Immediate Instruction			
001000	00001	00010	0000000101011110
OP Code	Addr 1	Addr 2	Immediate value

Equivalent mnemonic: **addi \$r1 , \$r2 , 350**

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

# A64 Instruction Set

## C4.1 A64 instruction set encoding

The A64 instruction encoding is:



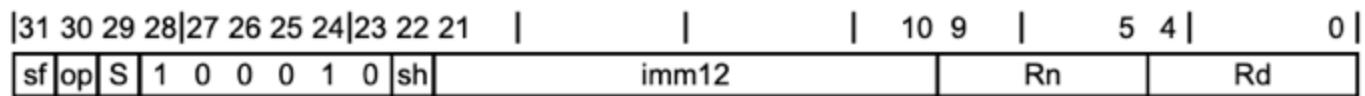
**Table C4-1 Main encoding table for the A64 instruction set**

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



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

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




---

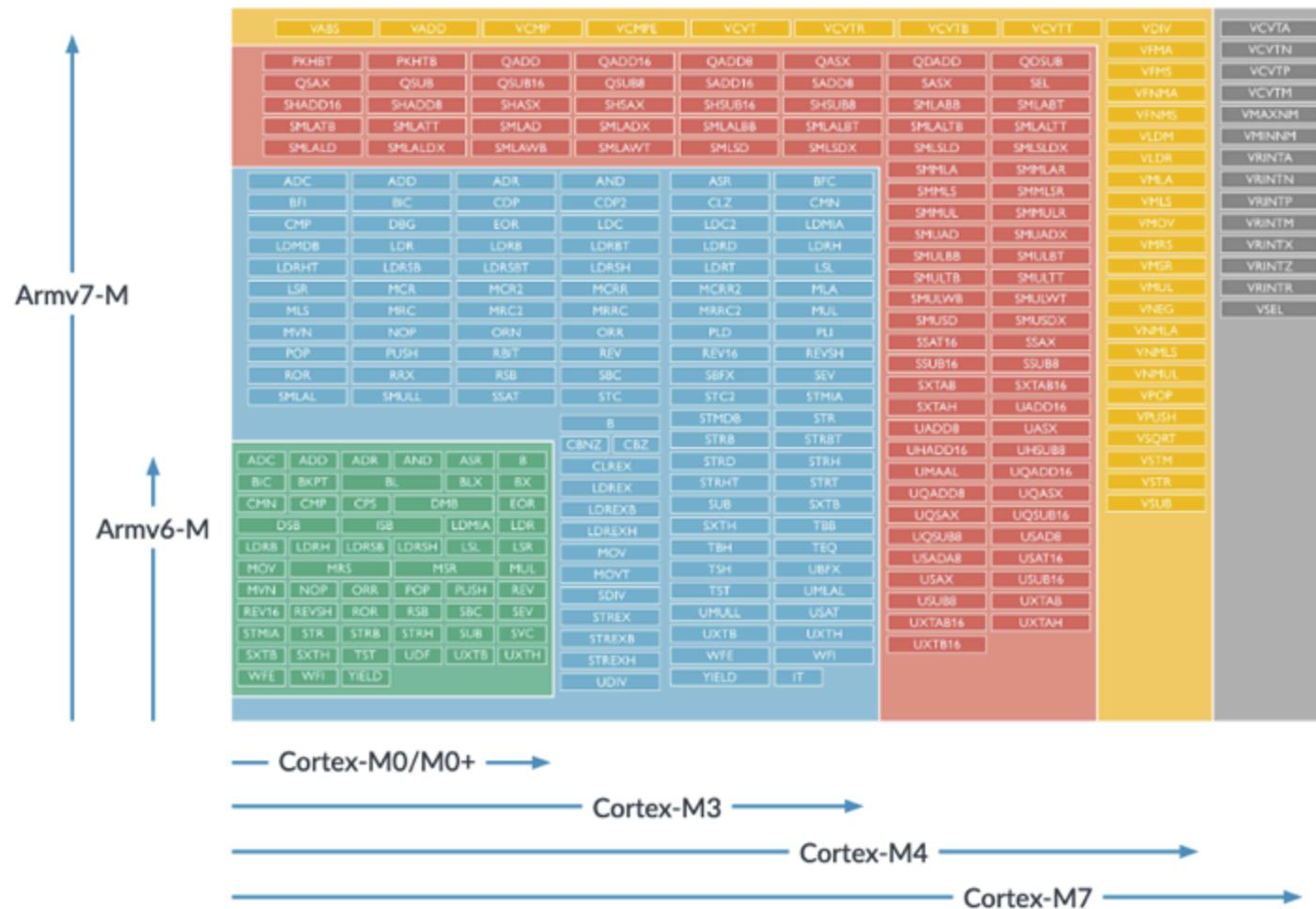
#### Decode fields

#### Instruction page

sf	op	S	Instruction page
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



## 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**

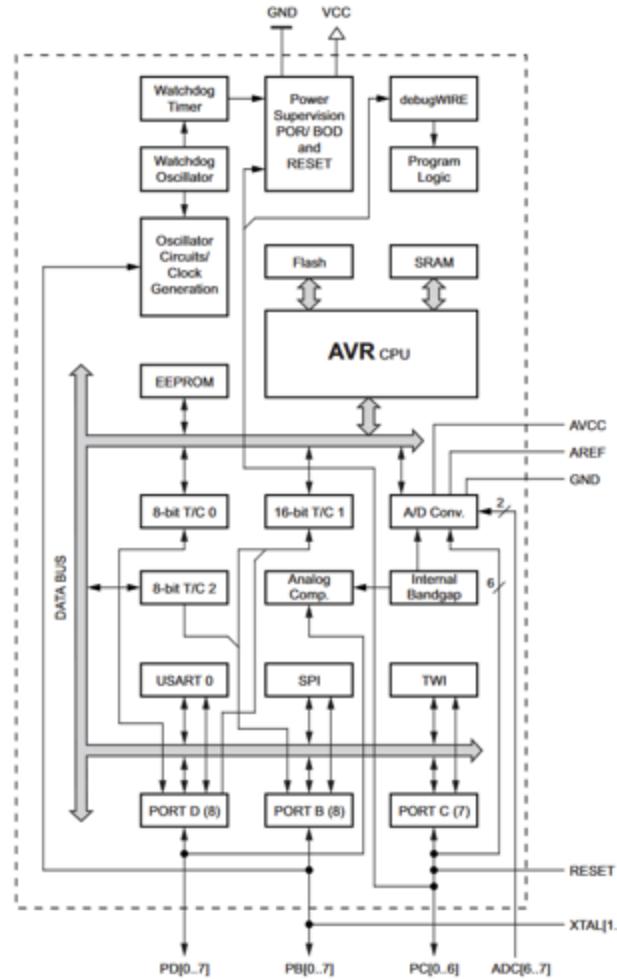


## Reduced Instruction Set Computer (RISC)

- Besser geeignet für "moderne" Compiler
- Intel / AMD haben lange den CPU Markt mit CISC CPUs dominiert
- Im mobile und embedded Bereich ist ARM (RISC) extrem verbreitet
- Seit 2020 gibt es auch im Desktop wieder RISC Systeme (Apple M1) mit grossen Vorteilen in der Effizienz
- Verschiedene Hersteller bieten auch für RISC Server-CPUs an die v.a. bei Cloud Anbietern (AWS, Google, etc) Verbreitung finden

# **SoC vs Microprocessor vs Microcontroller**

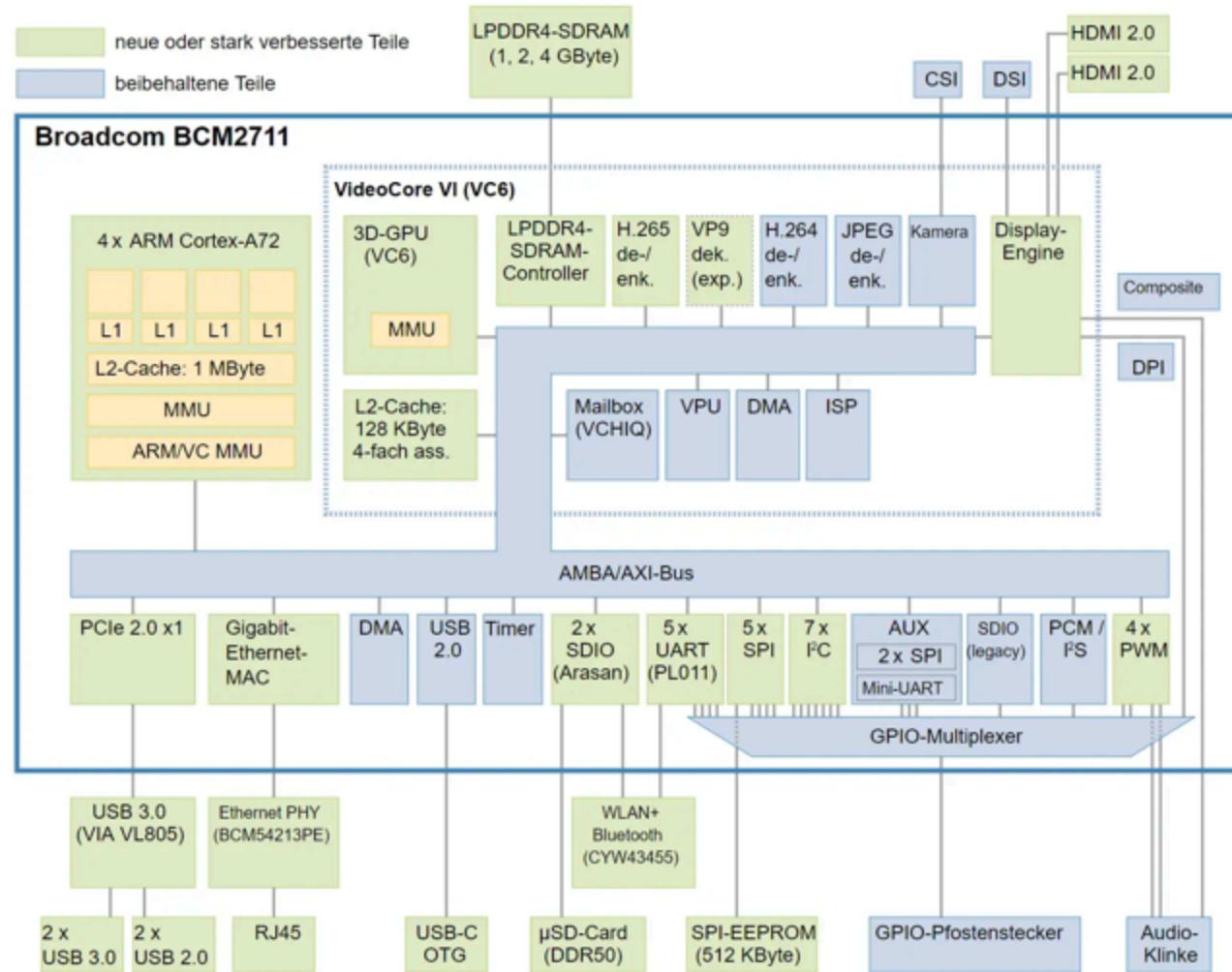
# Microcontroller: ATmega328P

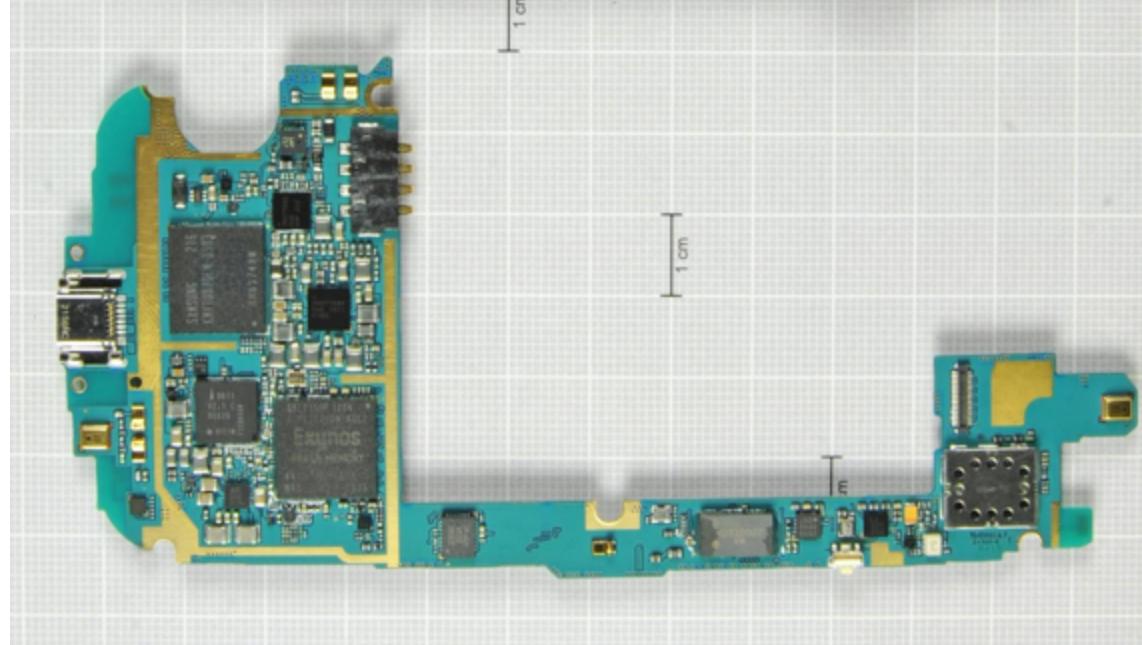


# System on Chip (SoC)

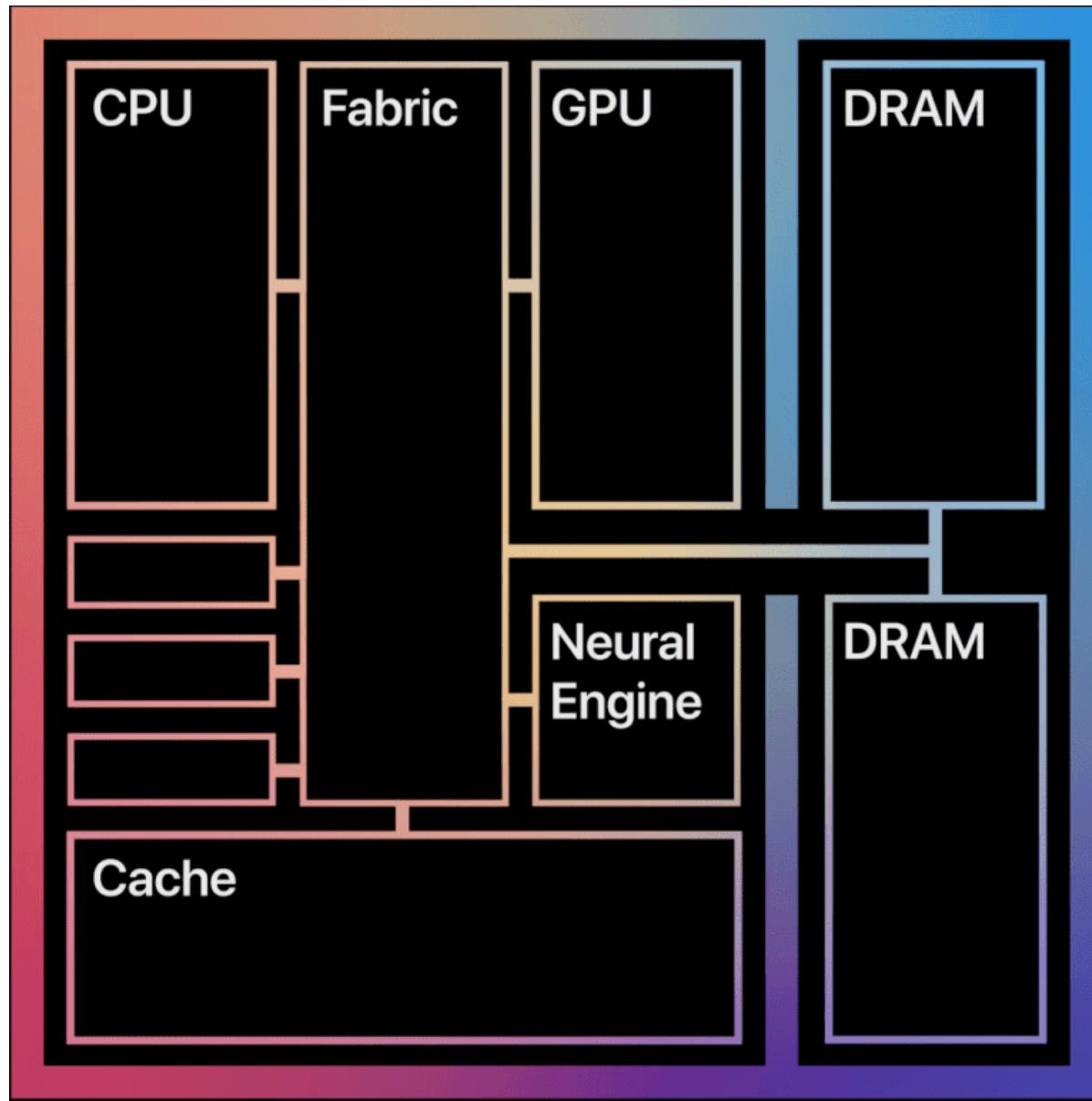
## Herz des Raspberry Pi 4: Broadcom BCM2711

Das System-on-Chip (SoC) BCM2711 vereint nicht nur vier CPU-Kerne mit einer GPU, sondern enthält auch Controller für viele Schnittstellen.



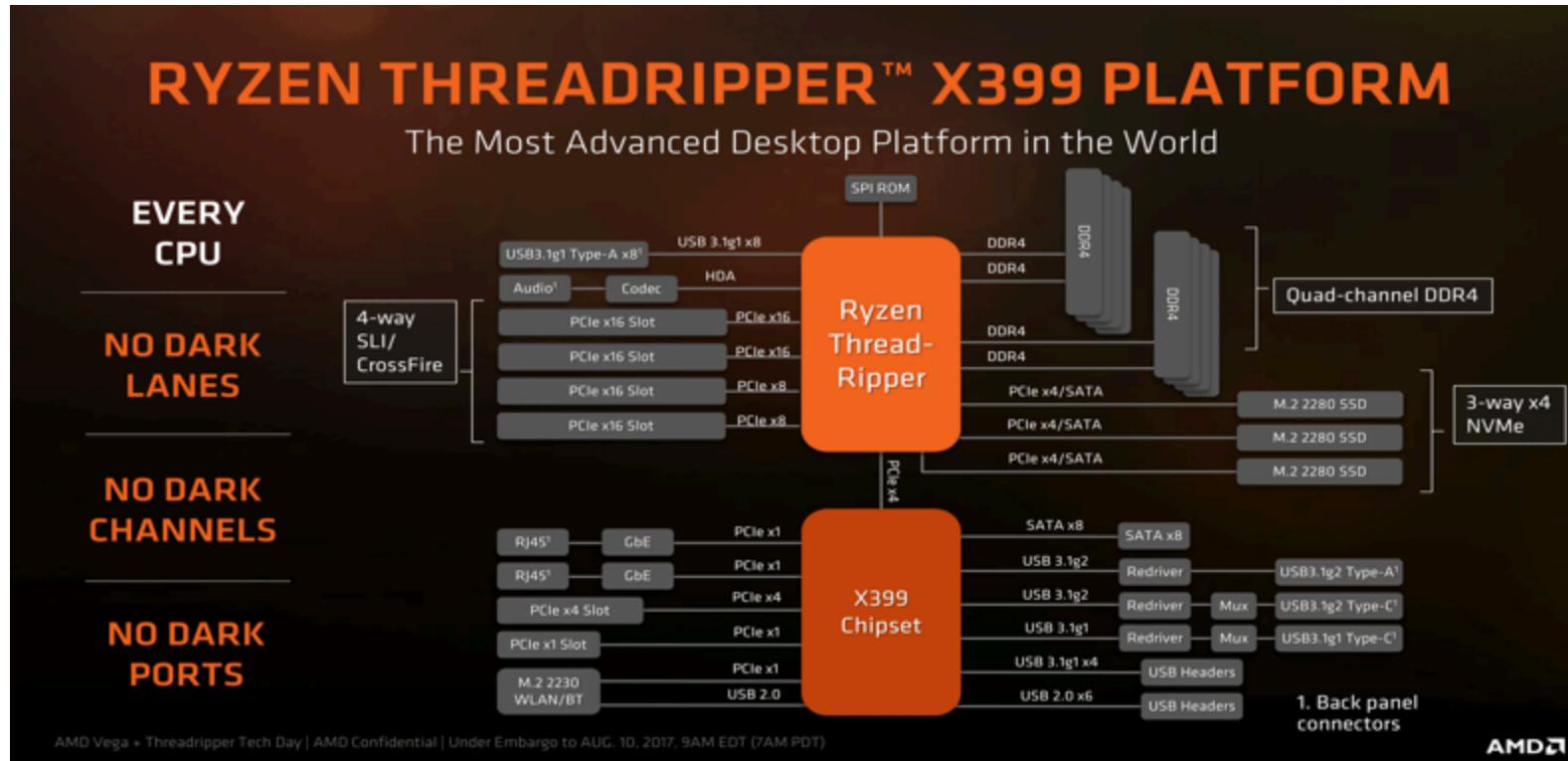


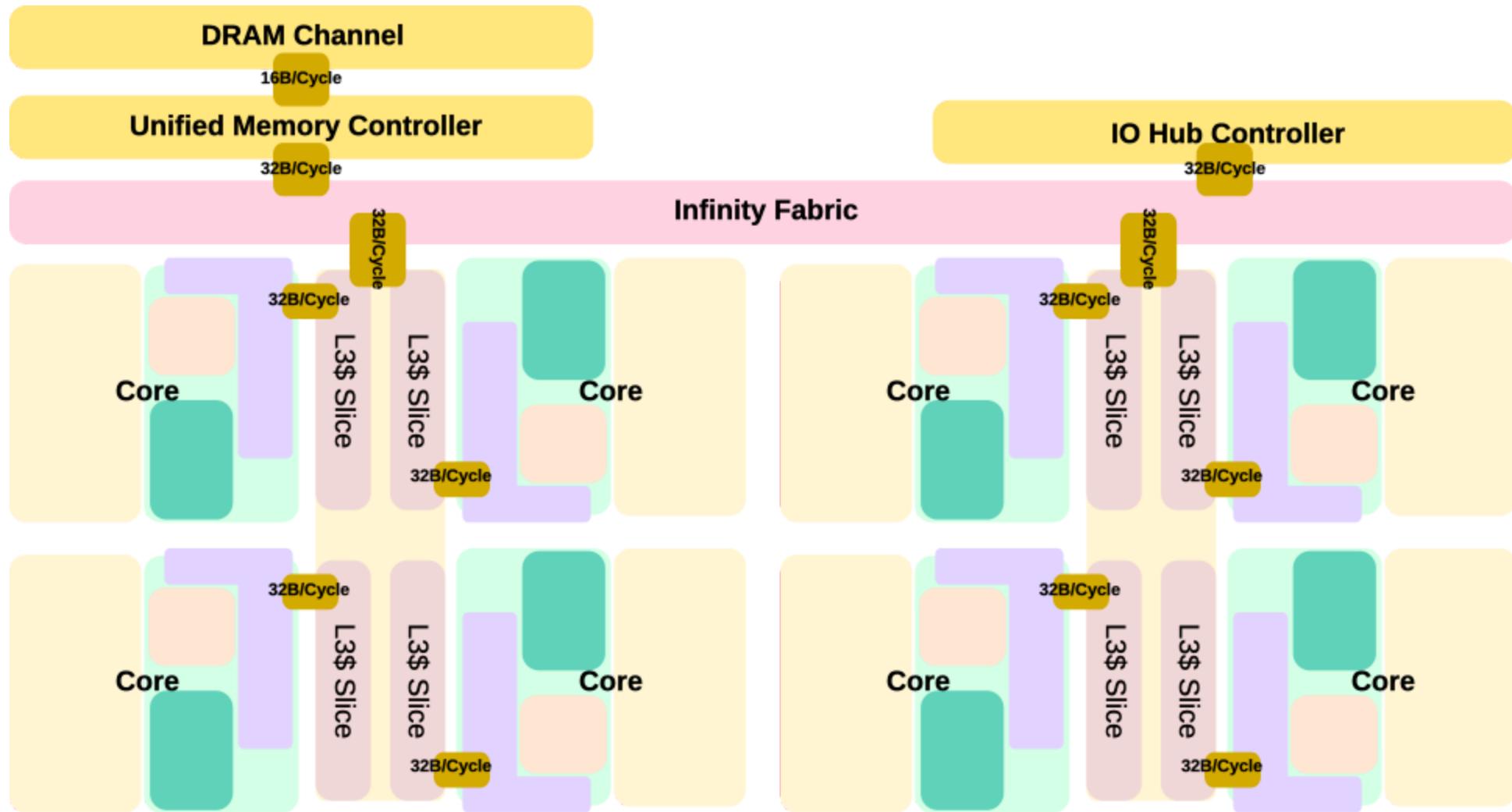
Samsung Galaxy S3



Apple M1

# Microprocessor: AMD Ryzen Threadripper





## **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."

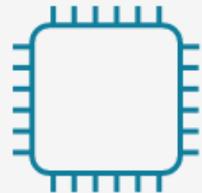
([https://group.softbank/en/ir/financials/annual\\_reports/2021/message/segars](https://group.softbank/en/ir/financials/annual_reports/2021/message/segars),  
08.01.2024)

## Company Highlights



**70%**

of the world's population  
uses Arm-based  
products



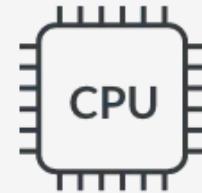
**270Bn+**

Arm-based chips shipped  
to date



**99%**

of smartphones run on  
Arm-based processors



**50%**

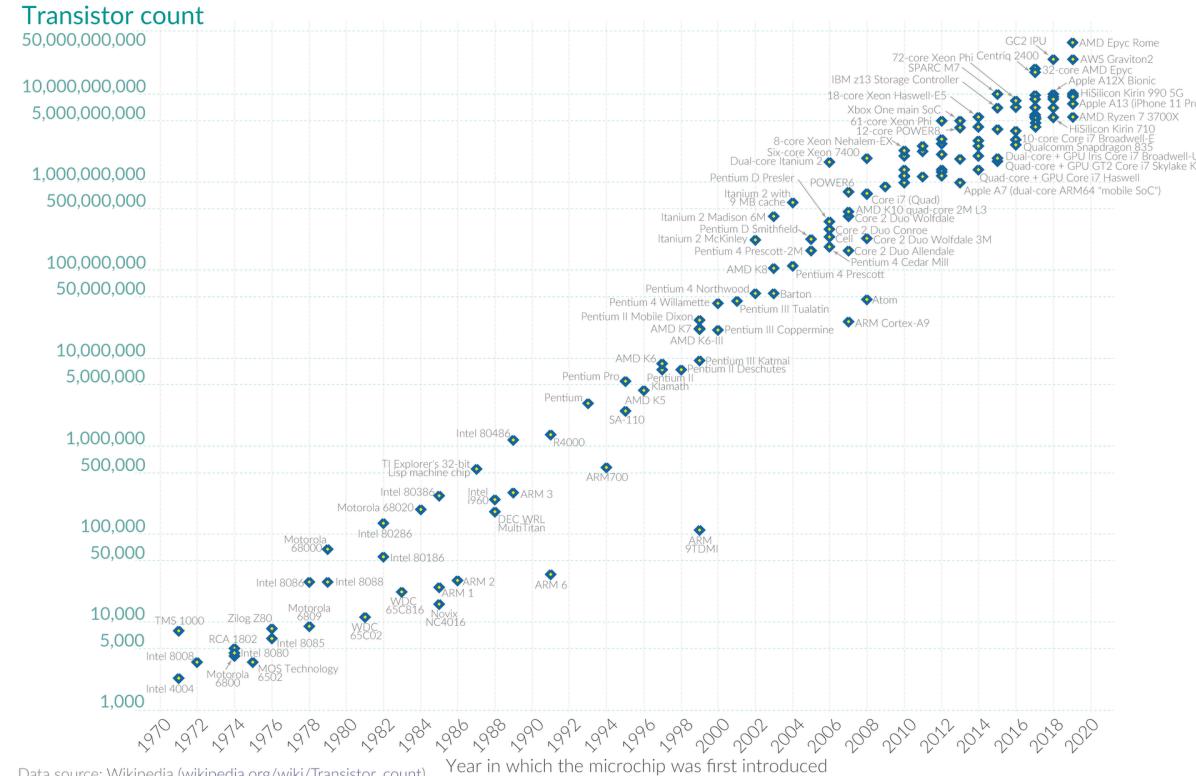
of all chips with  
processors are Arm-  
based

# Moore's Law

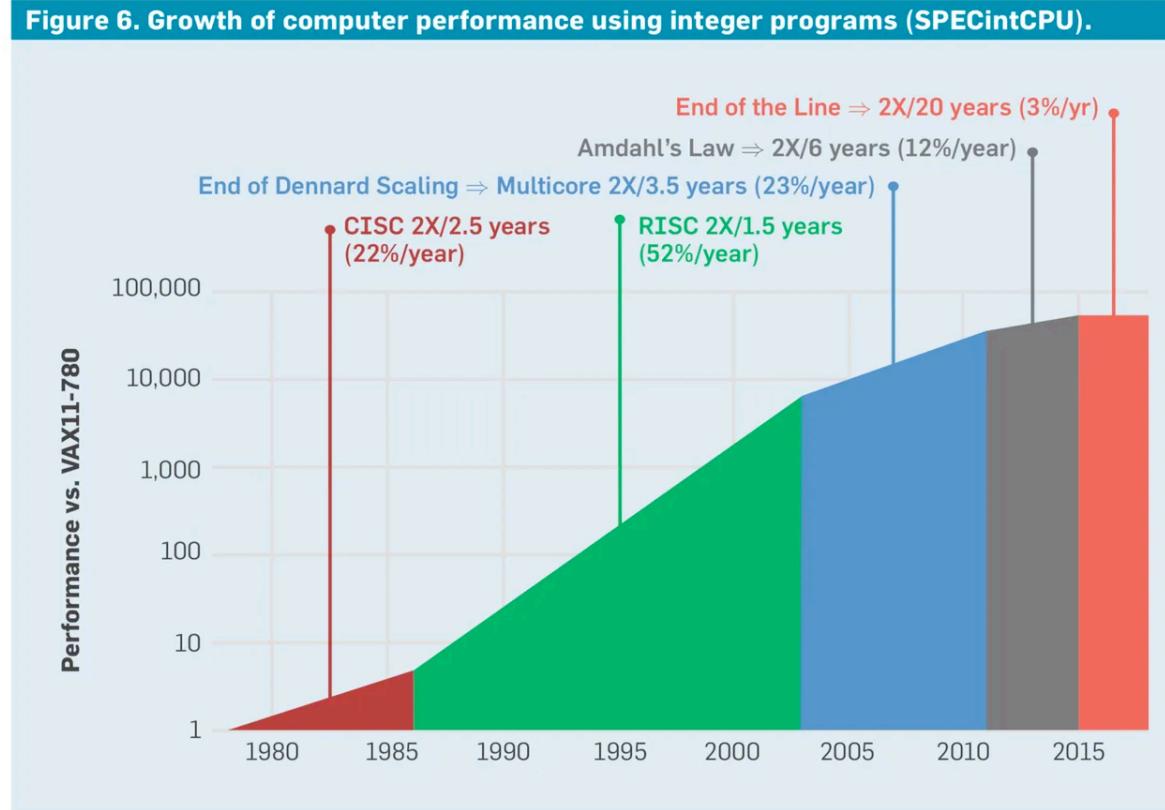
Moore's Law: The number of transistors on microchips doubles every two years

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

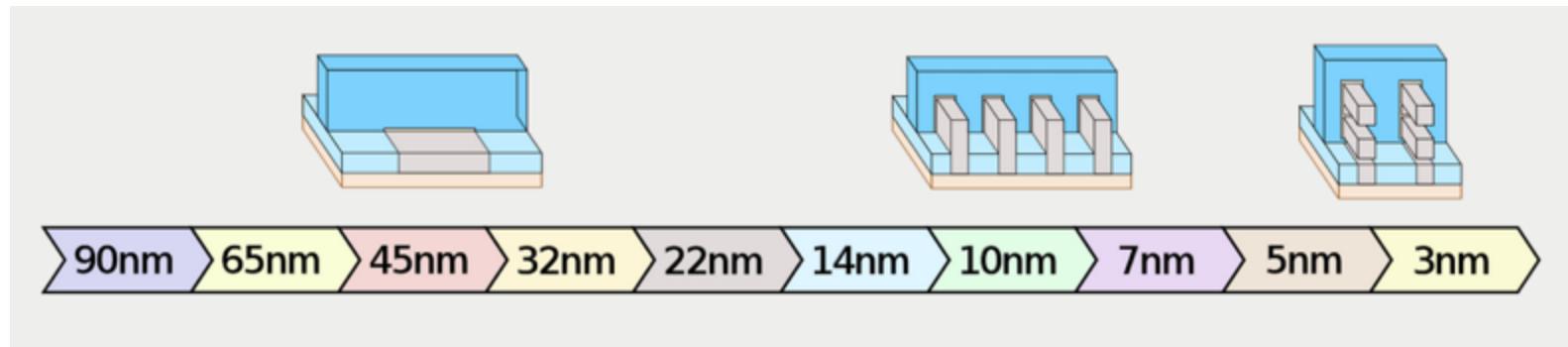


**Figure 6. Growth of computer performance using integer programs (SPECintCPU).**

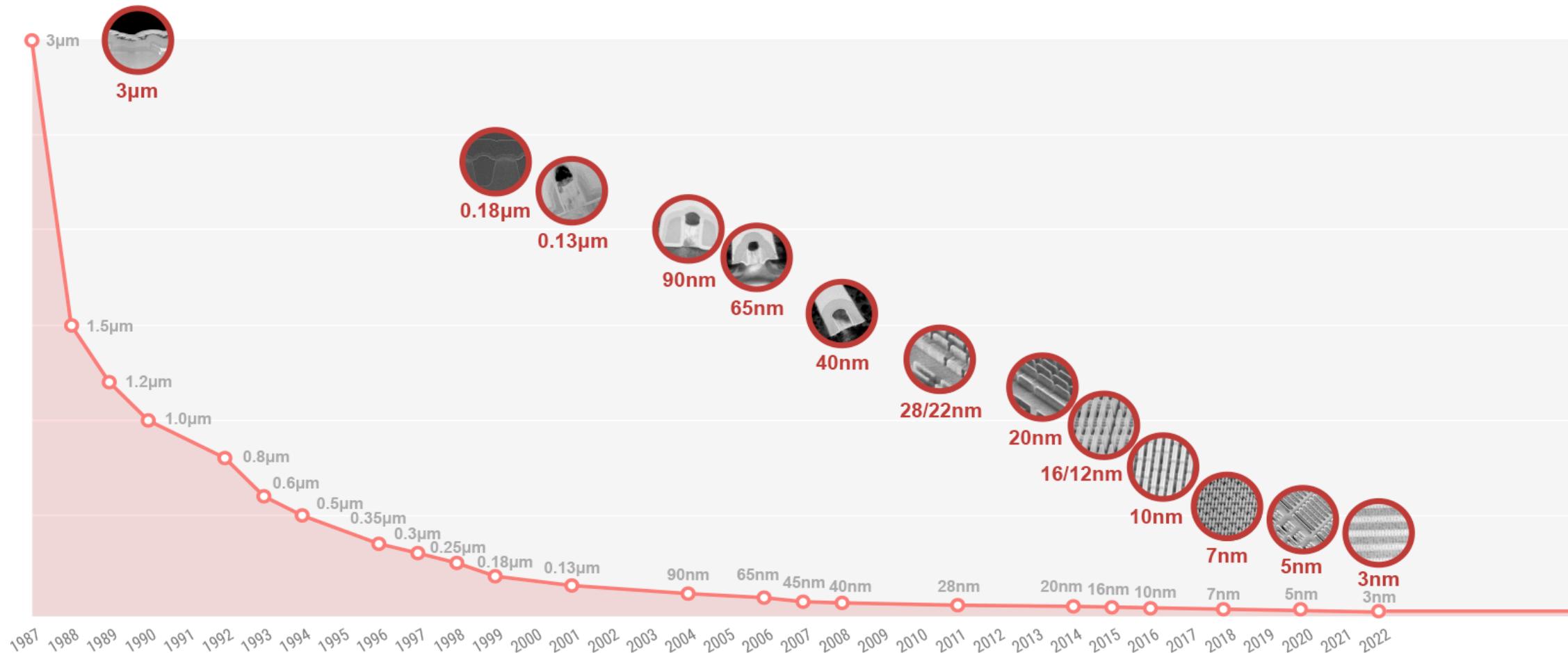


<https://www.zdnet.com/article/ai-is-changing-the-entire-nature-of-compute/>  
(Patterson, Hennessy, 2014, S.44)

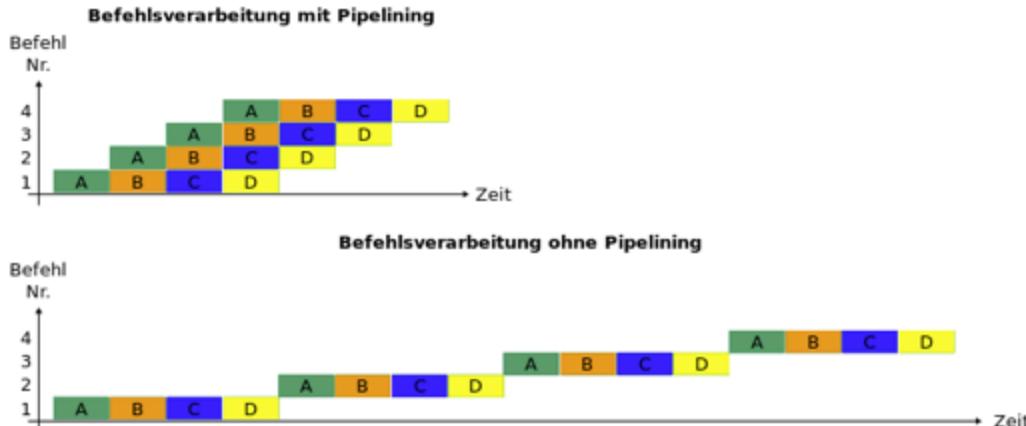
# Strukturgrösse



# TSMC



# 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)

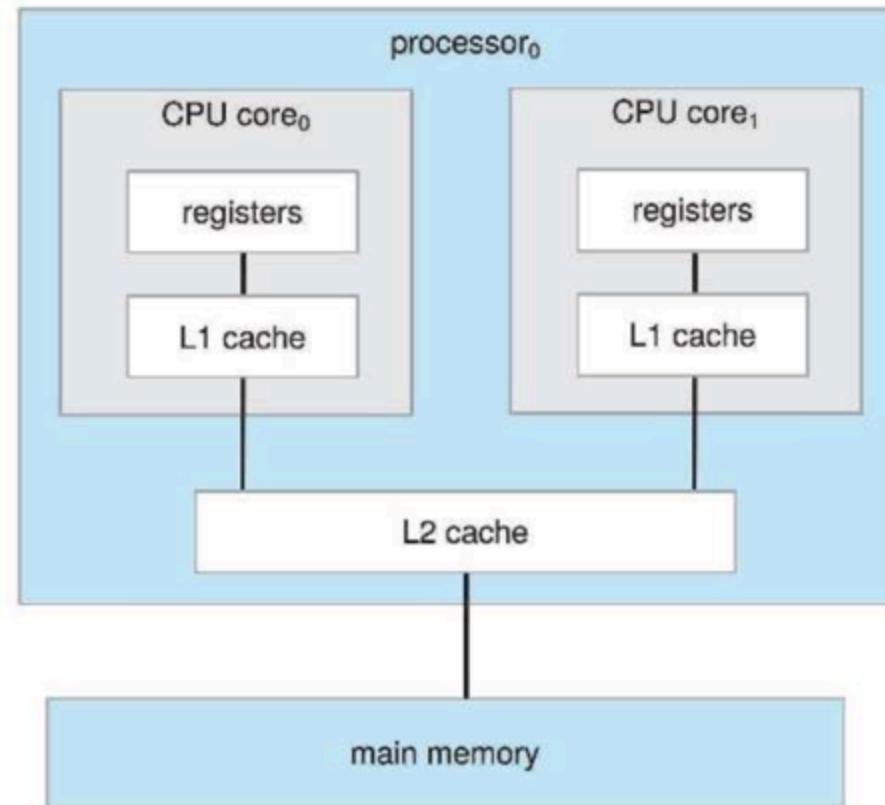
In der Ausführungsphase wird der dekodierte Befehl ausgeführt. Das Ergebnis wird durch den Pipeline-latch gepuffert.

## D – Ergebnisse zurückgeben (WB, Write Back)

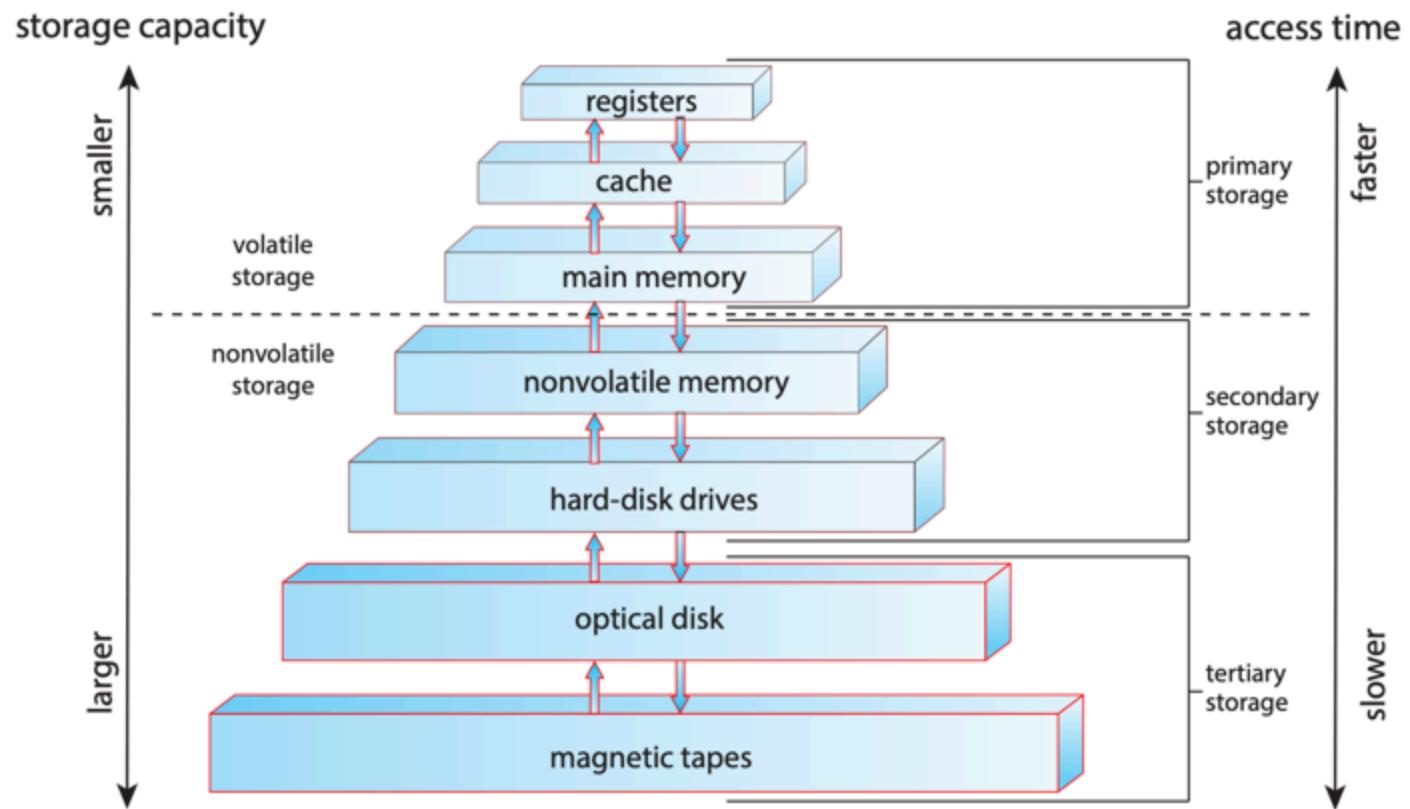
In der Resultatspeicherphase wird das Ergebnis in den Arbeitsspeicher oder in den Registersatz zurückgeschrieben.

# **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)

## Zahlendarstellung und Datentypen

- Binäre Zahlen: Für Maschinen einfach darstellbar (2 mögliche Zustände, idR. Spannungen)

## **Integer**

- Ganze Zahlen
- Natürliche Zahlen (Negativ): Das MSB (most significant bit) wird für das Vorzeichen verwendet

## Fliesskommazahlen

- Normiert in IEEE 754

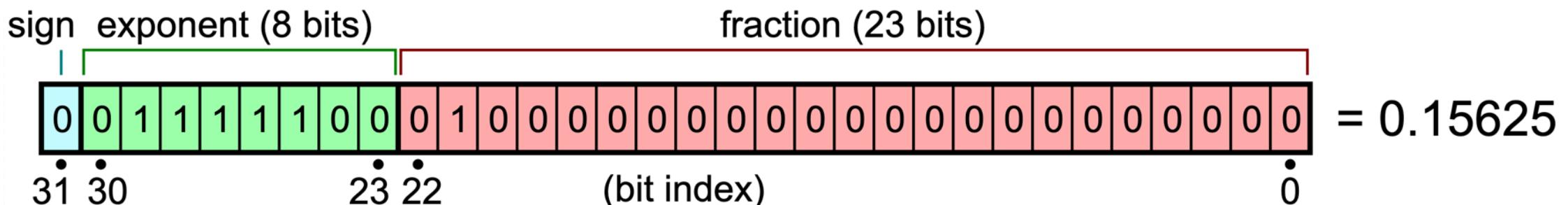
- $x = s \cdot m \cdot b^e$

- Vorzeichen s

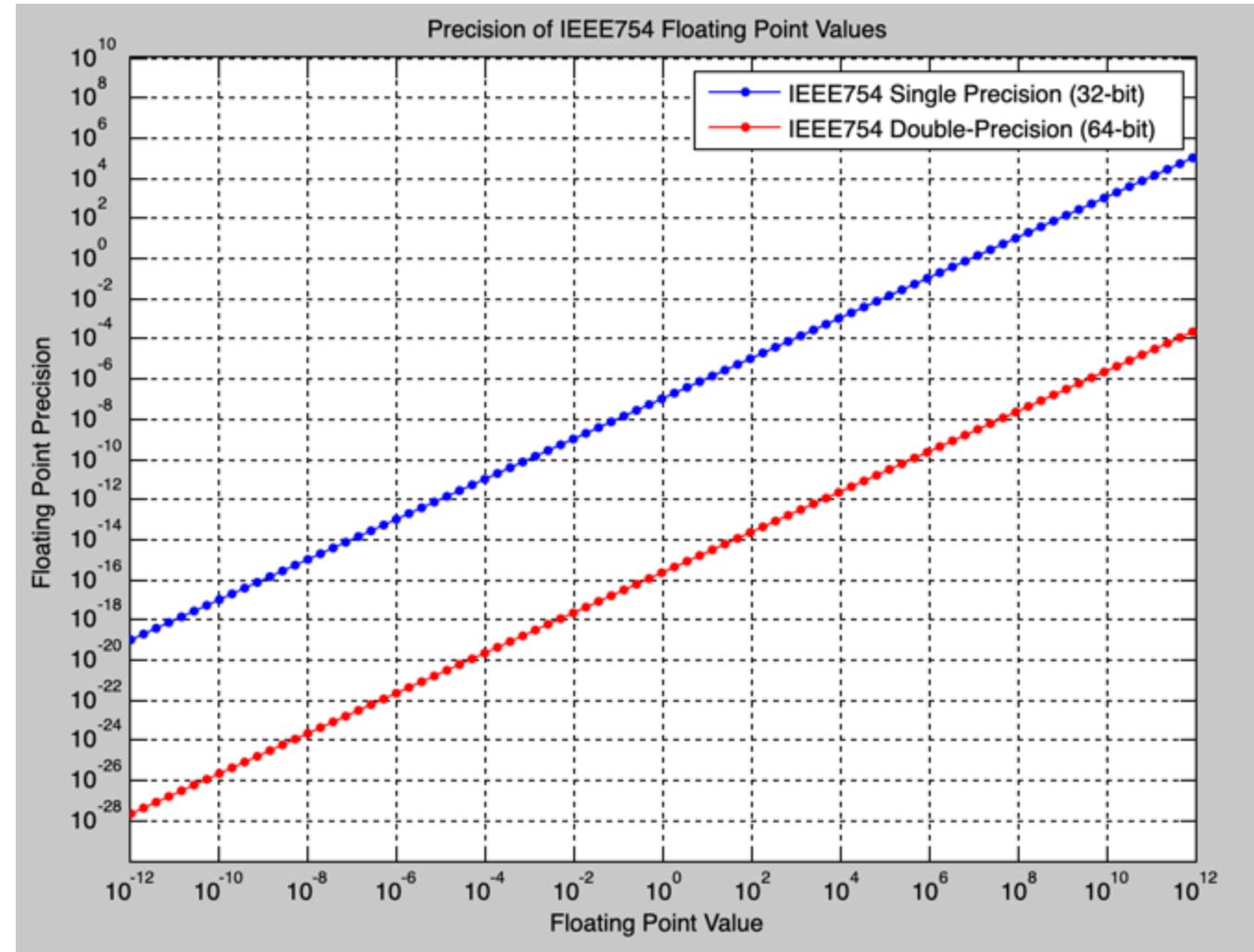
- Mantisse m

- Basis b (b=2)

- Exponent e



# Floating Point: Präzision



# Strings

- Array von Buchstaben (Char)

## ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[END OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	/	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

## Datentypen in Go (Auswahl)

`bool` boolean, 1-bit, true or false

`int8` 8-bit signed integer (-128 bis 127)

`int16` 16-bit signed integer (-32'768 bis 32'767)

`int32` 32-bit signed integer (-2'147'483'648 bis 2'147'483'647)

`uint8` 8-bit unsigned integer (0 bis 255)

`float32` 32-bit IEEE 754 floating-point number (1.2E-38 bis 3.4E38)

`string` "Sequence of Unicode code points"

# Statische Typisierung

- Zur Laufzeit hat jedes Objekt einen (Daten)typ
- Im Programmtext hat jeder Ausdruck einen Typ → Der Typ ist zum Zeitpunkt der Kompilierung bekannt
- Vorteile
  - Fehler können früher erkannt werden
  - Effizientere Programme, da keine Typprüfung während der Laufzeit
  - Mehr Optimierungsmöglichkeiten durch Compiler
- statisch typisierte Sprachen: Java, Kotlin, C#, C, Go, Rust

```
final Crossroad crossroad = new Crossroad();
final CrossroadController crossroadController =
final Scene scene = new Scene(crossroadControll
```

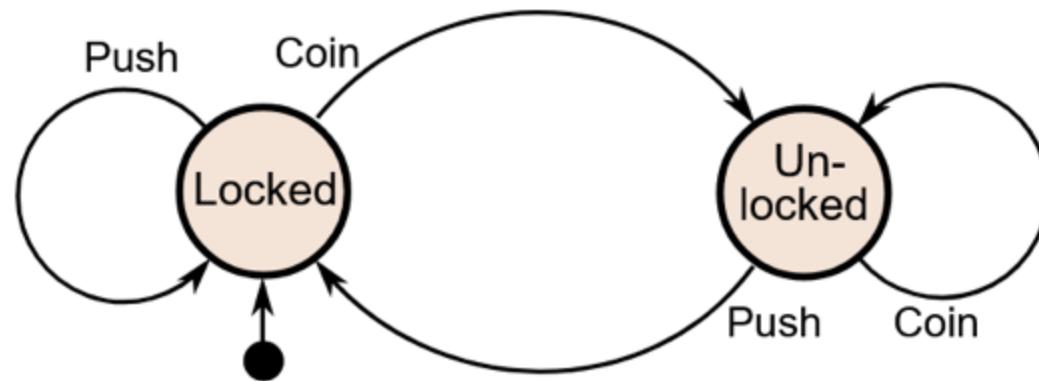
## Datentypen in Python (Auswahl)

- `str`
- `int` (Kein Limit)
- `float` (64Bit IEEE 754))
- `complex`
- `bool`

## Dynamische Typisierung

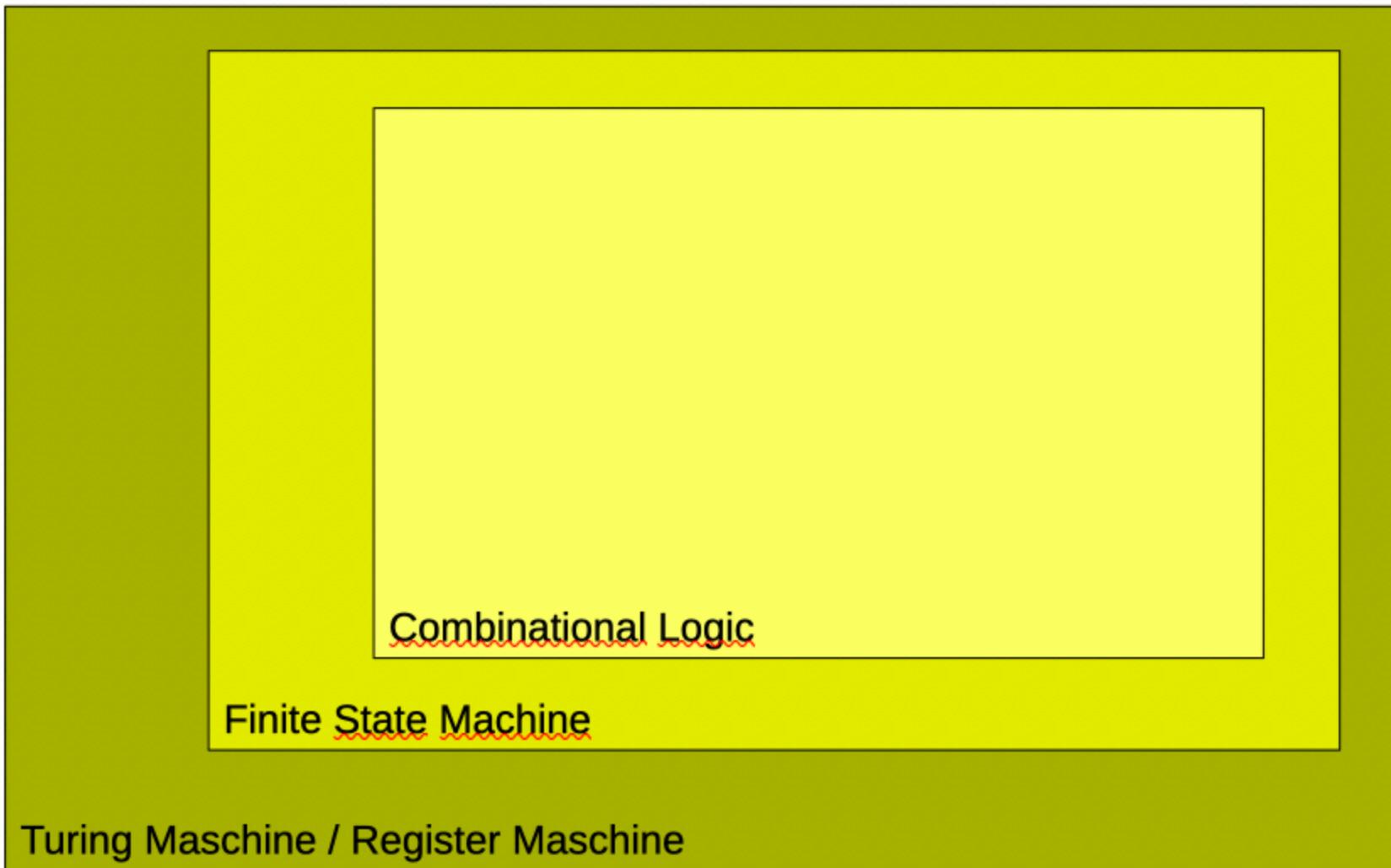
- Zur Laufzeit hat jedes Objekt einen Typ
- Der Typ wird zur Laufzeit geprüft
- Duck Typing: "When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."
- Vorteile
  - Einfachere Programmierung
- Durch Typehints kann die IDE uns bei der Entwicklung dennoch unterstützen
  - `def greeting(name: str) -> str:`
- dynamisch typisierte Sprachen: PHP, Python, Ruby, JavaScript

# Finite State Machine

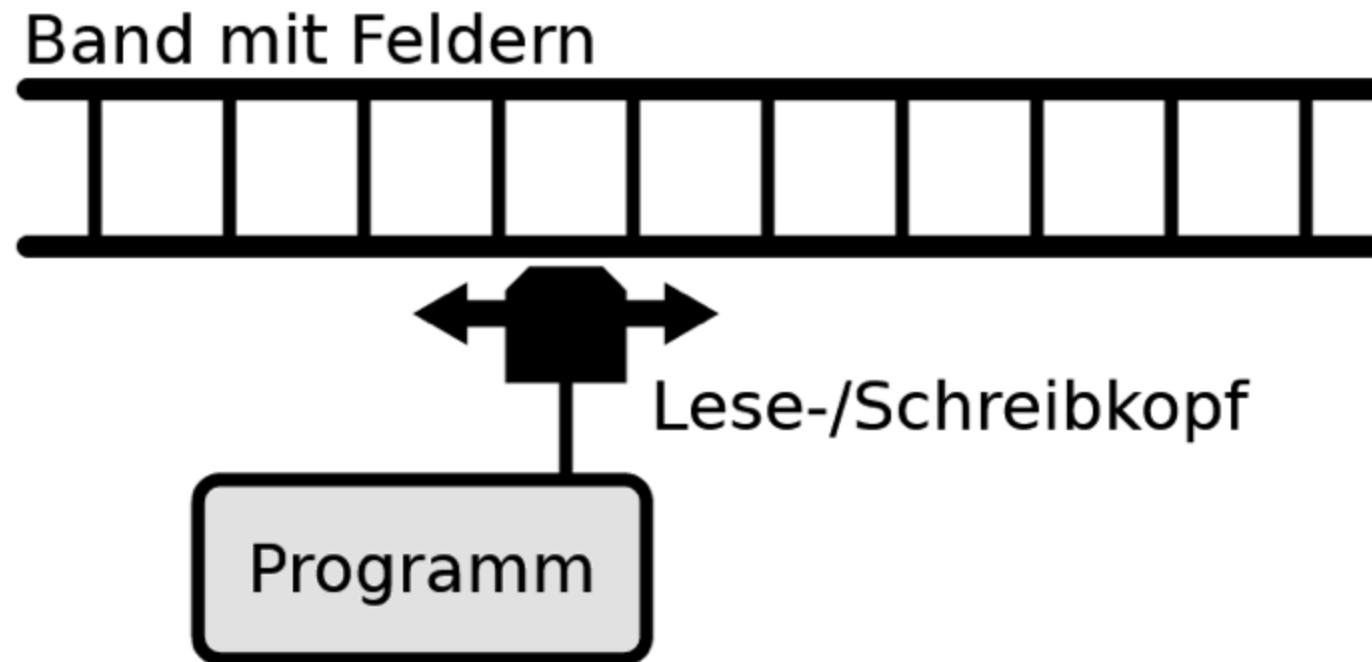


Current State	Input	Next State	Output
Locked	coin	Unlocked	Unlocks the turnstile so that the customer can push through.
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When the customer has pushed through, locks the turnstile.

# Automatentheorie



# Turing-Maschine



## Quellen

Silberschatz, 2019

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

Patterson, Hennessy, 2014

: D.A.Patterson, J.L.Hennesy (2014): Computer Organization and Design - The Hardware / Software Interface, Fifth Edition, Morgan Kaufmann