Microcontroller Techniques

Winter Term 2016/2017

Prof. Dr. L. M. Reindl

Laboratory for Electrical Instrumentation
Department of Microsystems Engineering – IMTEK
University of Freiburg

Your instructor

Prof. Dr. Leonhard Reindl

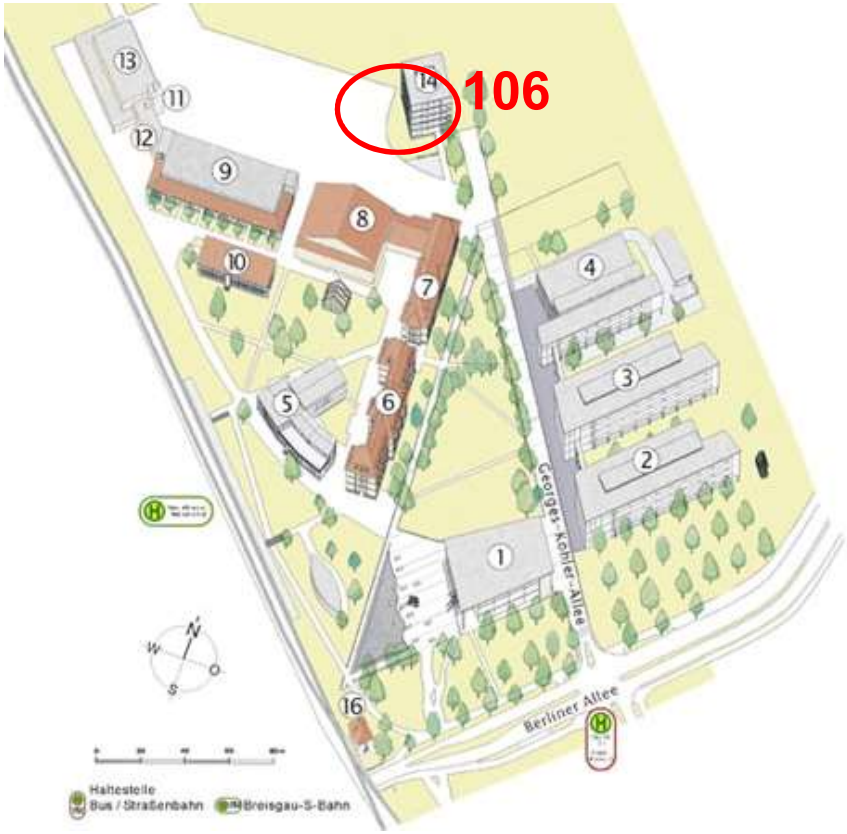
- 1985 Dipl.-Phys. at TU München
- 1985 cooperate research and technology (Siemens AG)
- 1997 PhD at TU Wien
- 1999 lecturer at TU Clausthal
- since 2003: professor and head of the Laboratory for Electrical Instrumentation at IMTEK, University of Freiburg

Contact Data

Prof. Dr. Leonhard Reindl

phone: 0761-203-7221
e-mail: reindl@imtek.de
office space: building 106, room 04-014

There are no set call times! Please make an appointment with the secretary (0761-203-7220) or write an e-mail.



Sebastian Stöcklin

contact person for lab course & exam
phone: 0761-203-7157
e-mail: sebastian.stoecklin@imtek.uni-freiburg.de
office space: building 106, room 04-004

ORGANISATIONAL

Learning content and scope of the lecture

The lecture and the practical course are just a teaser content of the field of microcontroller technology! It is not possible to treat these contents in full depth in one semester. Therefore, it is strongly recommended to rework the given topics with a textbook to obtain a deeper understanding.

Overview of the course

- **Lecture**
  - contents: theoretical basis & material collection about microcomputer theory
  - approx. 10 lectures (participation and rework recommended)
  - exam tutorial session(date & room: will be announced)
  - final exam - date & room: will be announced
- **Practical exercises**
  - contents: hands-on experience with microcontrollers (MSP430) & C programming
  - exercises will be performed at home, hardware available from the library of the technical faculty
  - 8 out of 9 exercise sheets must be passed with more than 50% of the points
  - introduction course on 26th of October (see e-mail)

Objectives of the course

- Basic understanding of modern data processing systems
  - digital logic
  - binary arithmetics
- Structure of microcontrollers (hardware)
  - microprocessor
  - peripherals
- Programming of microcontrollers controller (software)
  - (assembler language)
  - C
- Evaluation and selection of microprocessors
- Practical skills in systems programming in the laboratory course

Exam

Permitted aids

- Non-programmable calculator
- C command summary for programming task (provided by us)

The nature and extent of the exam

- approx. 7 to 10 tasks, 120 minutes of time.
- both theoretical tasks (based on the knowledge imparted in the lecture) and practical tasks (based on the exercises)

What's relevant?

- Basically, all contents of the lecture and the practical exercise are relevant!

Forums, training portal, etc.

- ILIAS: ilias.uni-freiburg.de
  Course „Microcontroller Techniques"
  - Password „ Lab20162017“
- Slides of the lecture, materials for the practical course, submission of exercise sheets, etc.
- Registration until 31th of October, 2016

Additional literature

- John Davies, „MSP430 Microcontroller Basics“, Springer, 2008.
- Matthias Sturm, „Mikrocontrollertechnik: Am Beispiel der MSP430-Familie“, Hanser, 2011.
- Marian Walter, Stefan Tappertzhofen, „Das MSP430 Mikrocontroller Buch“, Elektor-Verlag Aachen, 2011.
- Th. Flik., „Mikroprozessortechnik (RISC, CISC, Systemaufbau, Assembler und C)“, Springer, 1998.
- Beierlein Th., Hagenbruch O.: „Taschenbuch Mikroprozessortechnik“, Fachbuchverlag Leipzig, 2004.

Introduction

OVERVIEW OF THE LECTURE

Structure of the lecture

- Introduction

- Chapter 0: basic knowledge
  - number systems, digital logic, …

- Chapter 1: calculation and control systems
  - basic logic operations in CMOS, registers, memory, concepts, …

- Chapter 2: Basic peripherals
  - GPIOs, ADCs, DACs, …

Structure of the lecture

- Chapter 3: Advanced Peripherals
  - watchdog, programming interfaces, …
- Chapter 4: Programming
  - assembler language, C, compilers, …
- Chapter 5: Communication concepts
  - OSI model, topologies, UART, I2C, …
- Chapter 6: Outlook (among others: safe systems)

Introduction

MOTIVATION

Motivation

Why do we need microcomputers and -controllers?

These days, it is hardly conceivable to have equipment and processes without computer assistance. Microsystems technology provides sensors and actuators which are most useful with the help of computers, being integrated into the system to control and regulate a process.

Motivation

Forecasts characterized by terms such as

- Post-PC era
- Ubiquitous computing
- Pervasive computing
- Ambient intelligence
- Industry 4.0

© P. Marwedel, 2011

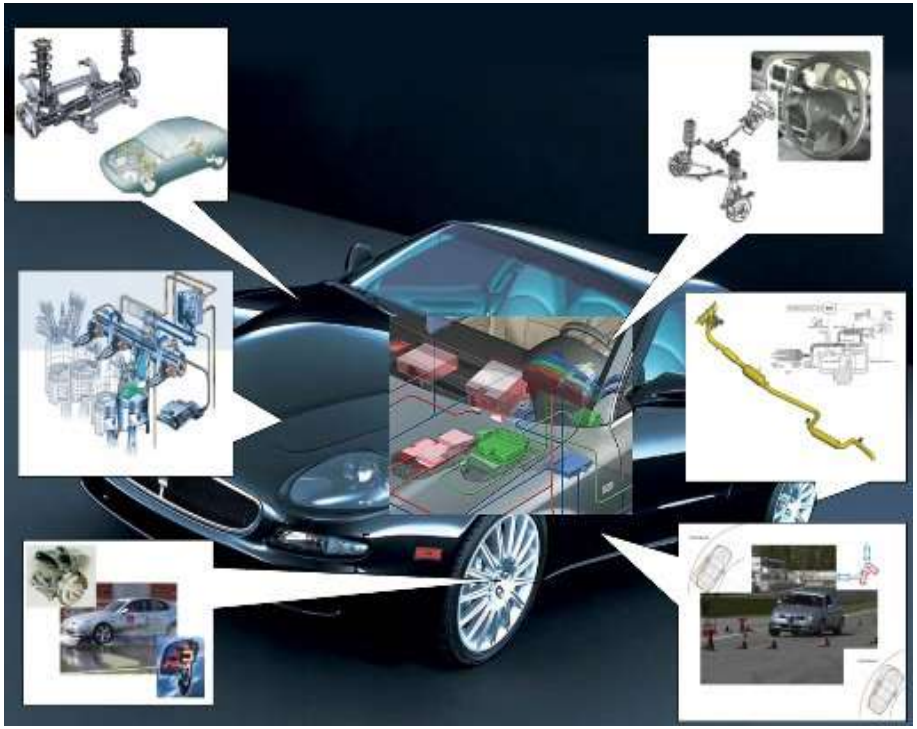
Embedded Systems

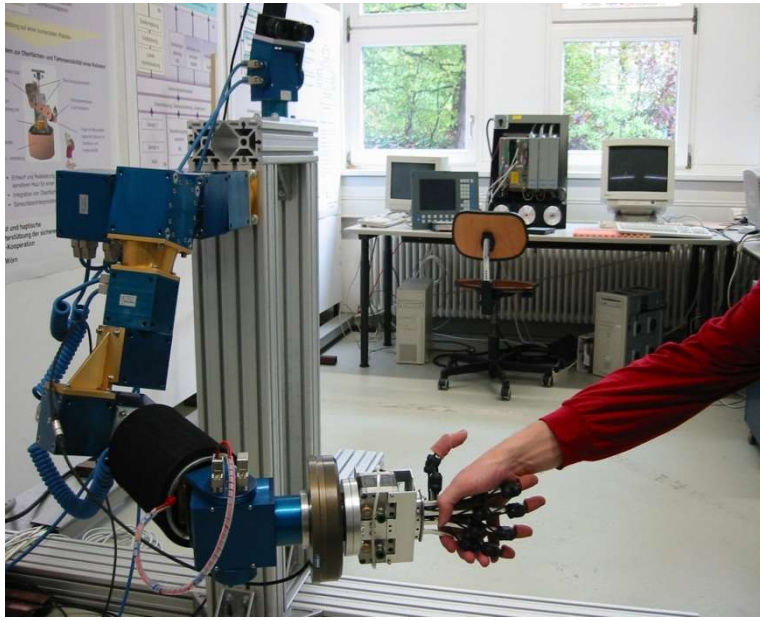
… are defined as information processing systems that

- are directly integrated into a surrounding product
- are controlling and monitoring this environment
- are adapted on the task, among other things in terms of
  - performance
  - power consumption
  - costs

Examples of embedded systems

Examples of embedded systems





Examples of embedded systems

Examples of embedded systems

Curiosity

- 256 MB DRAM
- 2 GB flash memory (to recognize errors and do trouble shooting)
- 256 KB EEPROM
- RAD750 CPU
  - Up to 200 MHz

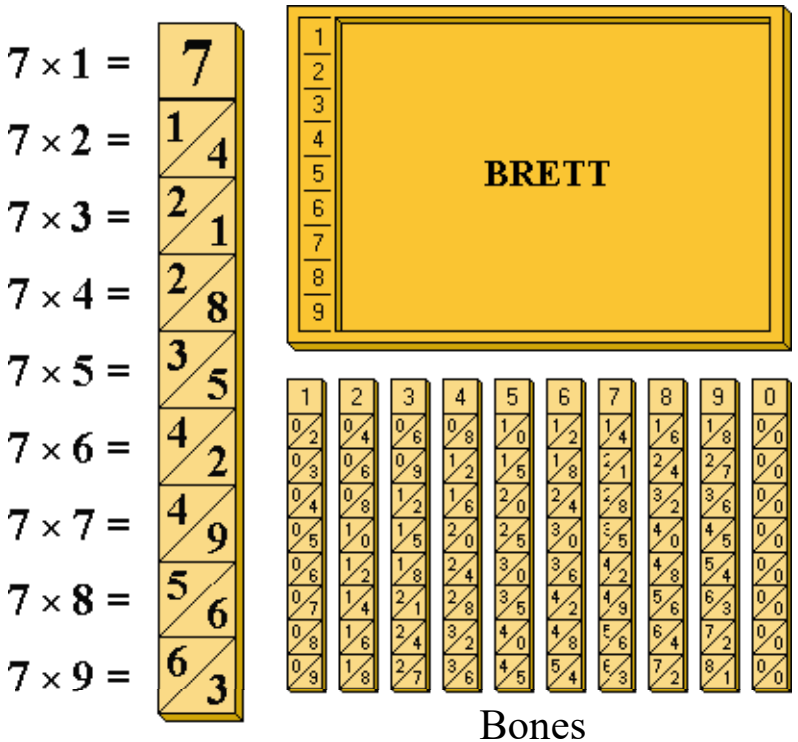
Introduction

HISTORY

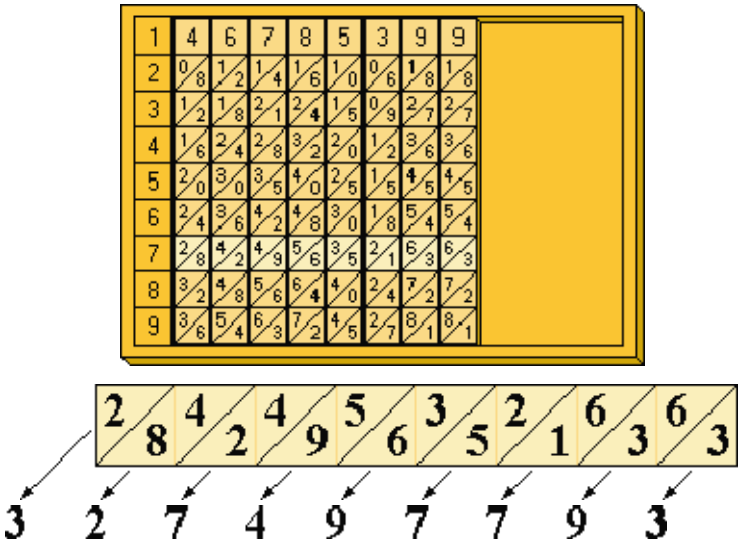
Basics

Adam Riese‘s "Auf den Linien“, which is equivalent to the Abacus

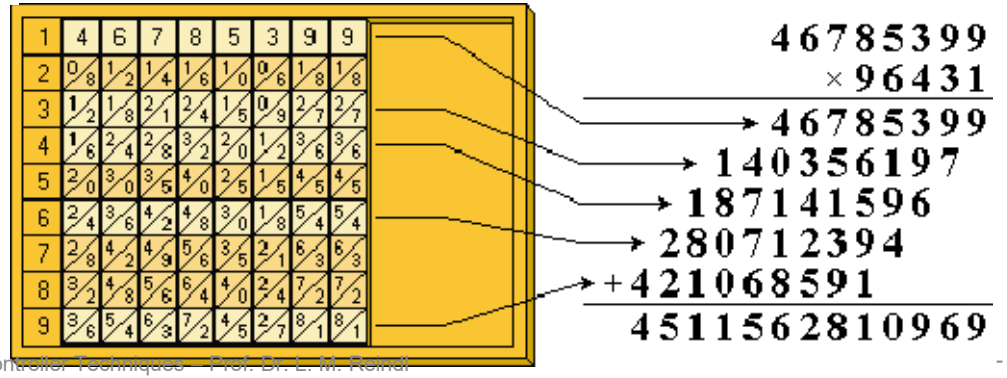
Basics - Napier's bones (1617)



Bones

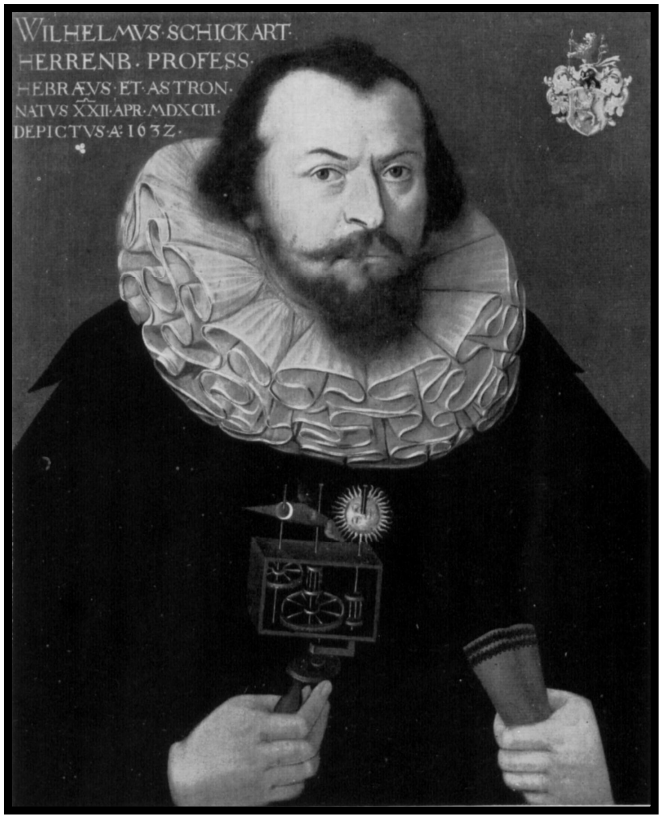


Computation: 7 x 46785399



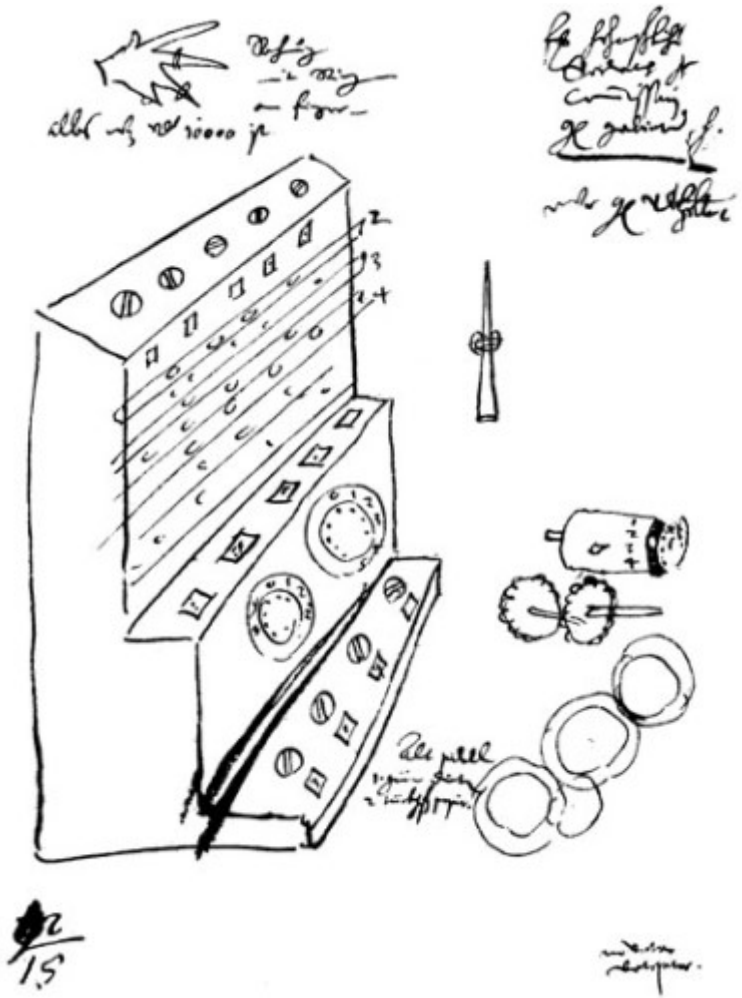
Basics - calculating clock

Replica of "calculating clock" of Schickard



Wilhelm Schickard 1592 - 1635

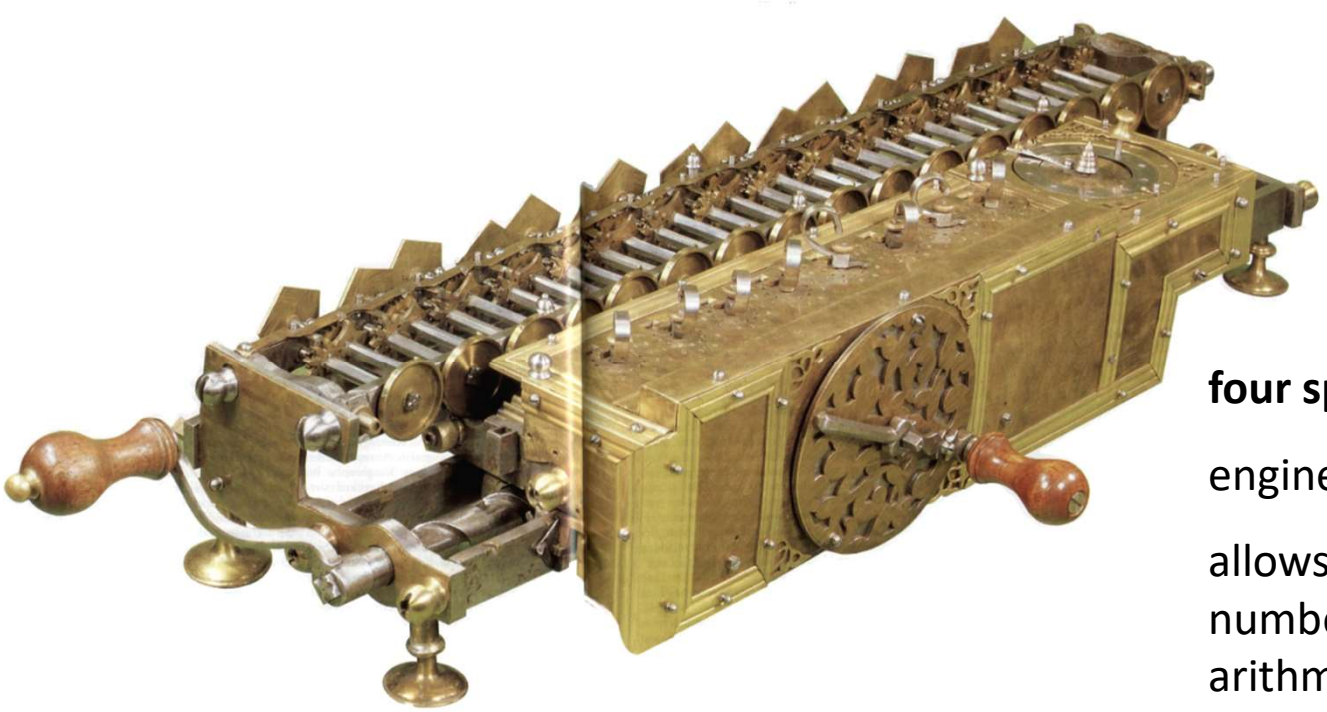
Basics - calculating clock



Original drawing of "calculating clock" of Schickard

- Allows summation or substraction of numbrers with up to six digits
- "Buffer overflow" indicated by ringing of a bell
- Napier's Bones for complex calculations

Basics - Four species Abacus



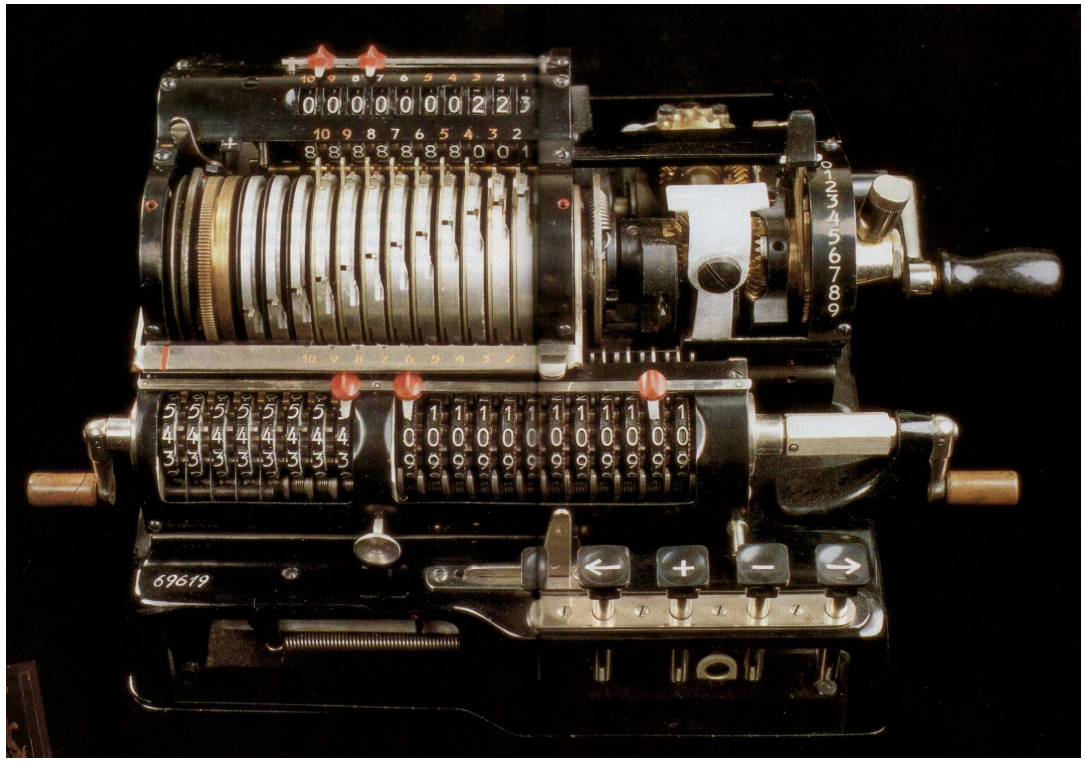
four species Abacus

engineered by Leibniz in 1673

allows the processing of 8-digit numbers using the four basic arithmetic operations addition, subtraction, multiplication and division

The machine failed during demonstrations in London because of manufacturing tolerances.

Basics - Brunsviga



Brunsviga Matador, 1905

Basics - Konrad Zuse

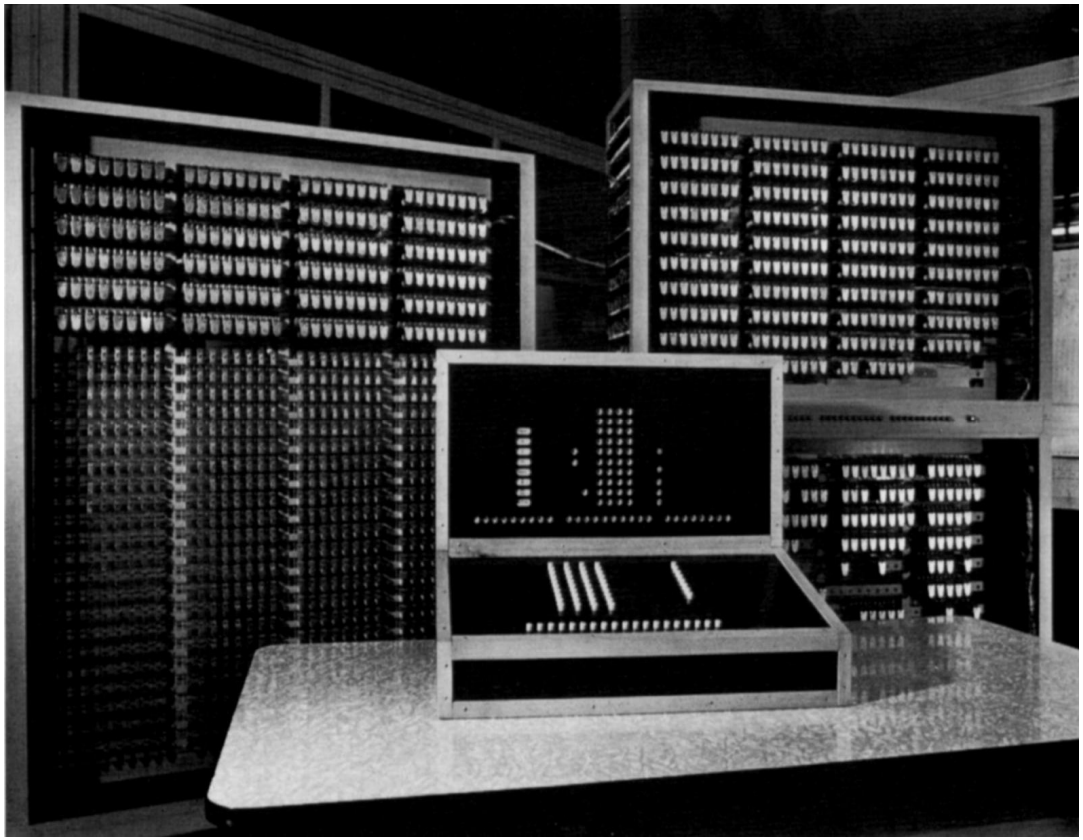
Konrad Zuse
* 22. June 1910 (Berlin)
† 18. December 1995

civil engineer

inventor of the (modern) computer

Zuse became aware of the works of Charles Babbage, Alan Turing or Howard Aiken not before the end of the Second World War.

Basics - Zuse Z3



Zuse Z3, Year 1940

first operable digital computer with programmability

Calculator: 600 relays
Memory: 1600 relays
calculates in dual system
floating point arithmetic
square root calculation lasts about 4 seconds

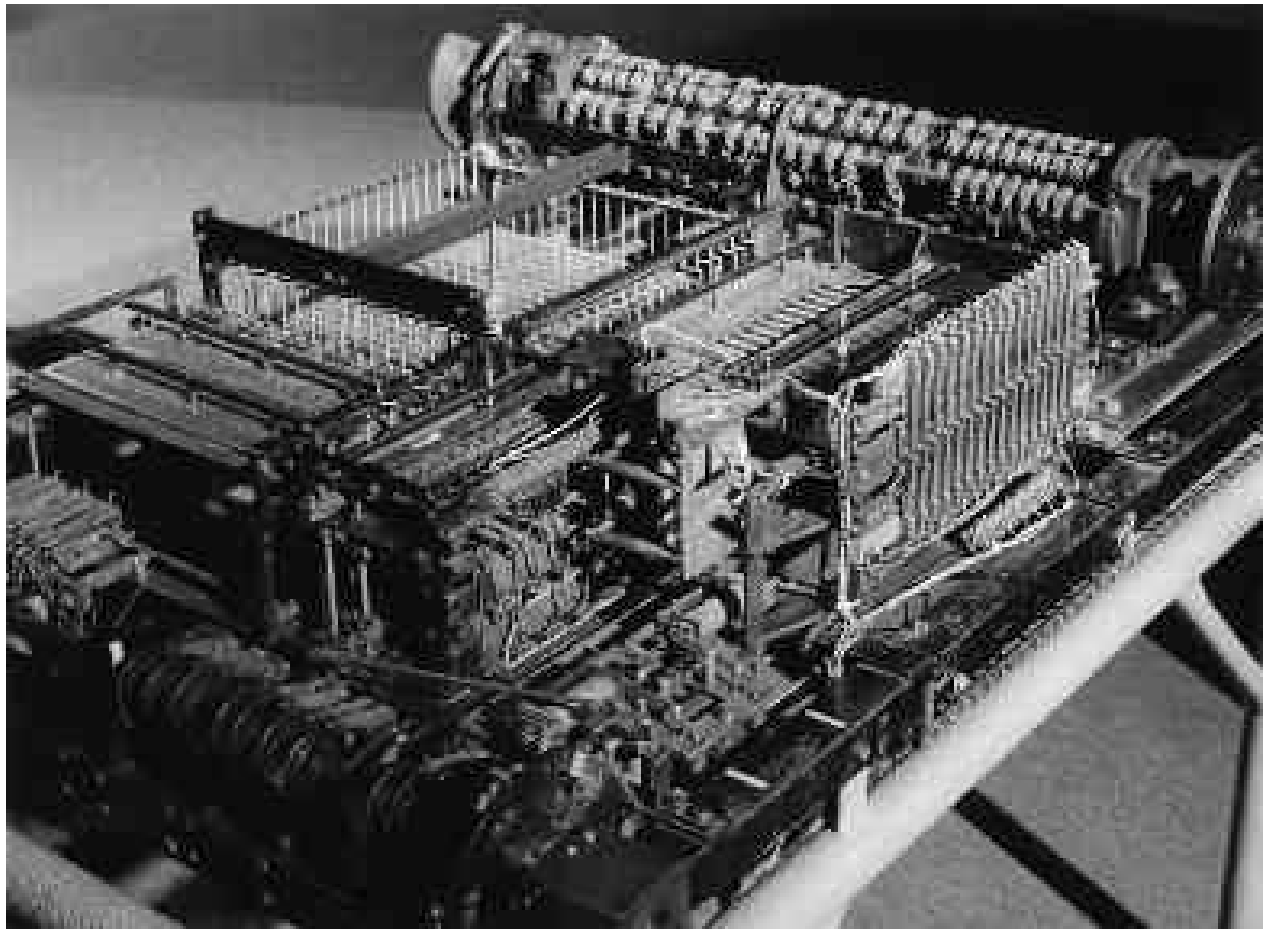
Basics - Zuse Z4

Zuse Z4, Year 1945

weight: 2,5 t
2200 relays total

4 kW of electric power

Basics – Zuse Z4



mechanical memory Z4
for 64 floating point words with 32 bit each

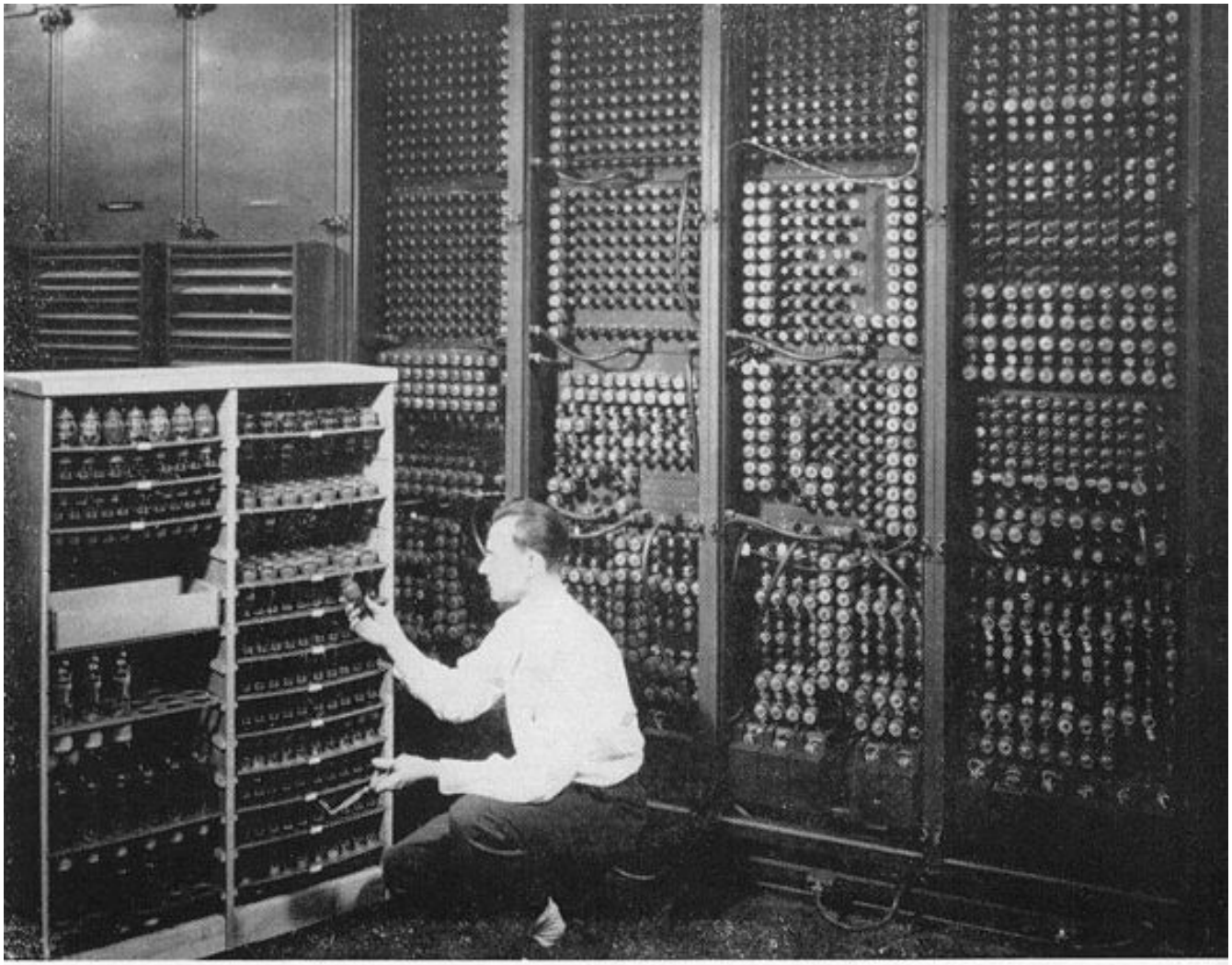
Basics - Z22

Z22, 1957, tube computer. addition 0,6 ms, multiplication 10 ms, still operable
(55 copies were sold; Image Museum of Technology Berlin)

Basics – ENIAC

- Electronic Numerical Integrator And Calculator
- In the early 40s (1944), the ENIAC was developed under the direction of J. Presper Eckert and John Mauchly at the University of Pennsylvania
- 17.468 vacuum tubes
- 30 tons
- 200 kW
- Performance equivalent to four species calculator (+, -, ·, :)
- clock speed of 100 kHz
- multiplication requires 3 ms
- Purpose of the development was the computer aided design of ballistics (e.g. calculating the trajectories of projectiles) for the U.S. Army in WW2.

Basics - ENIAC



Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

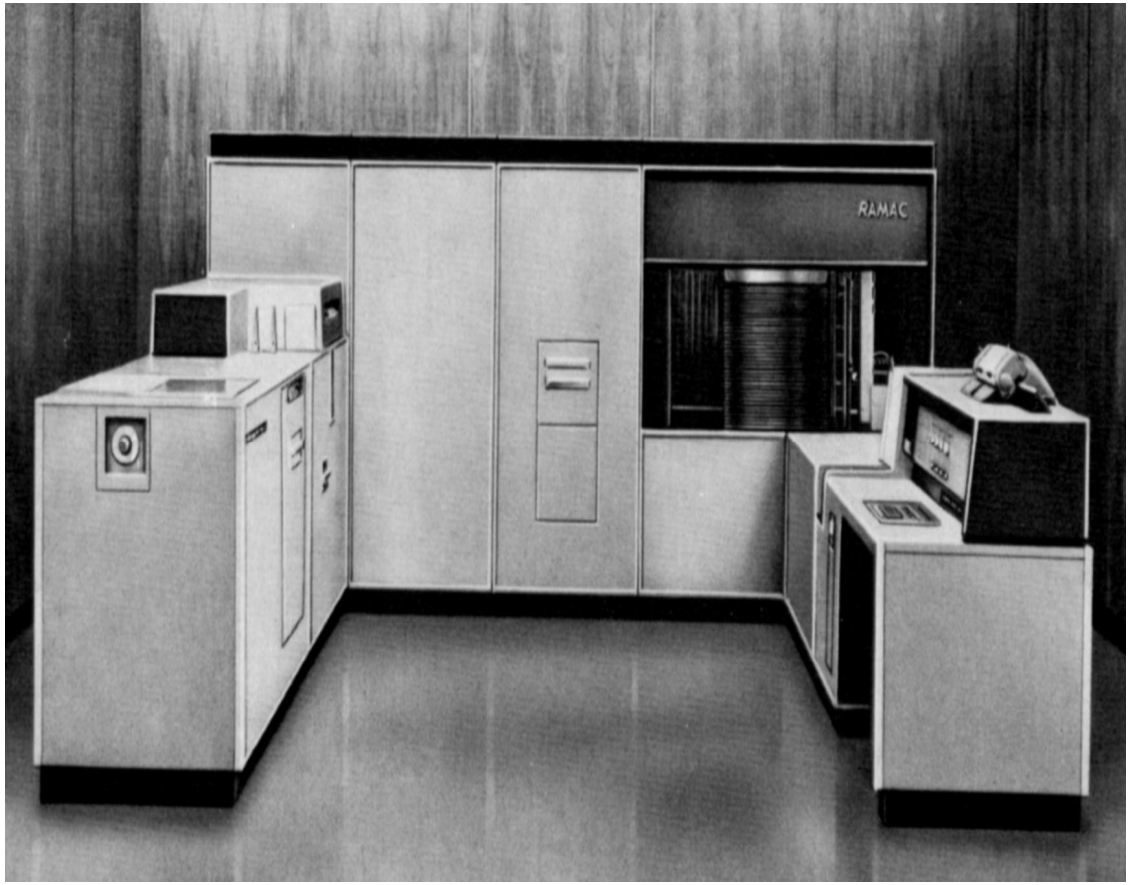
Basics – Mark I

Harvard Mark I, the first stored-program computing machine of the United States, 1944, 23-digit decimal numbers, 16 m long, 2.5 m high

Basics – Mark IV

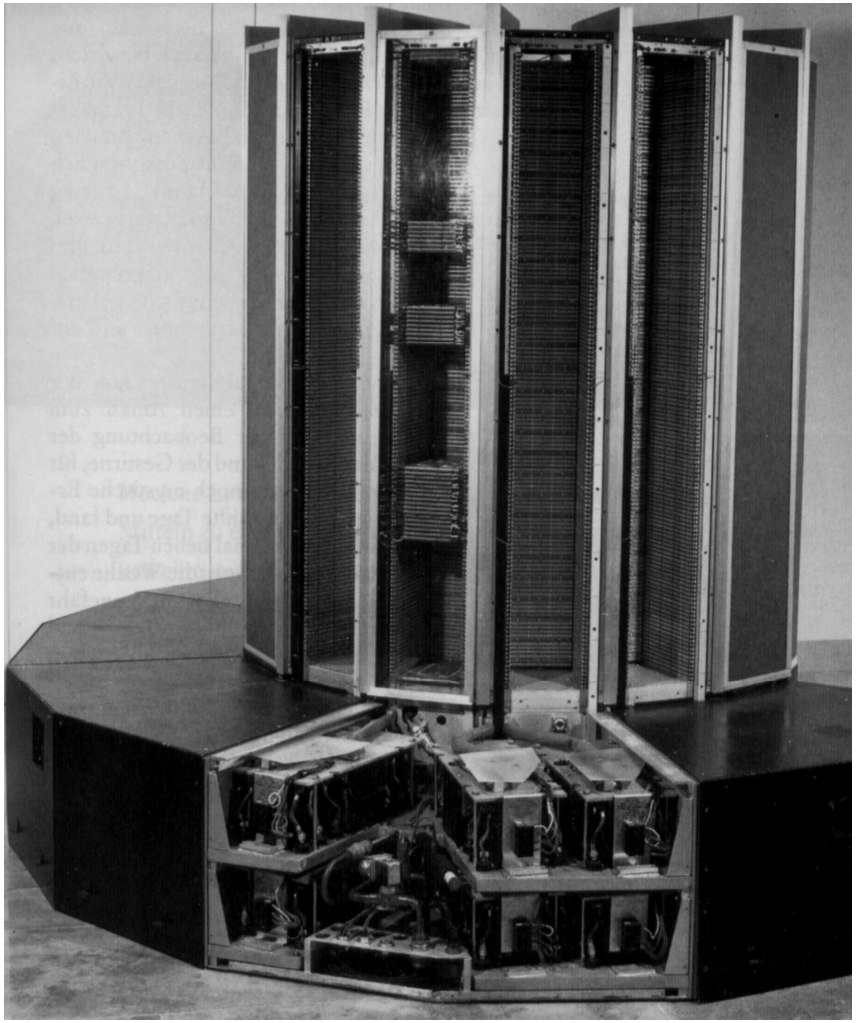
Harvard Mark IV, built by Aiken in 1952;
decimal computer with magnetic drum, 4000 cells, clock 16 kHz, 4000 tubes

Basics – IBM 305



IBM 305 "RAMAC", in use until 1963, 33 addtions per second, 3500 tubes, 1250 relays, ferrite core memory 100 places

Basics Cray

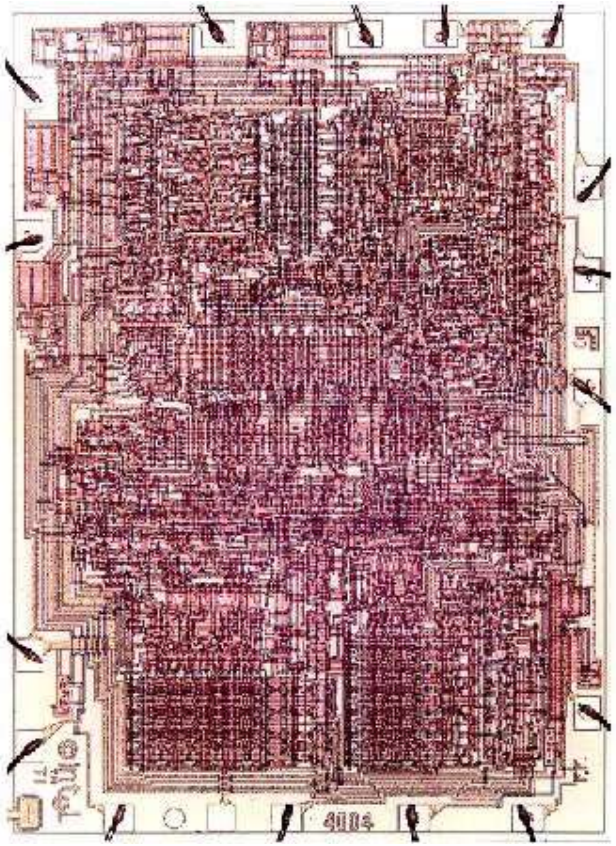


Cray-Supercomputer

first Version was launched in 1976

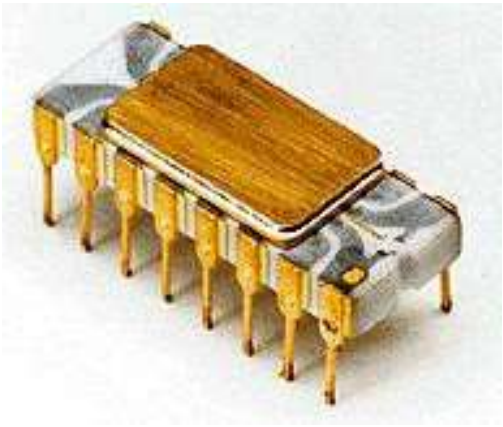
Intel 4004 (1971)

- 2300 Transistors
- word length of 4 bit
- up to 740 kHz



Intel Microprocessor Hall of Fame

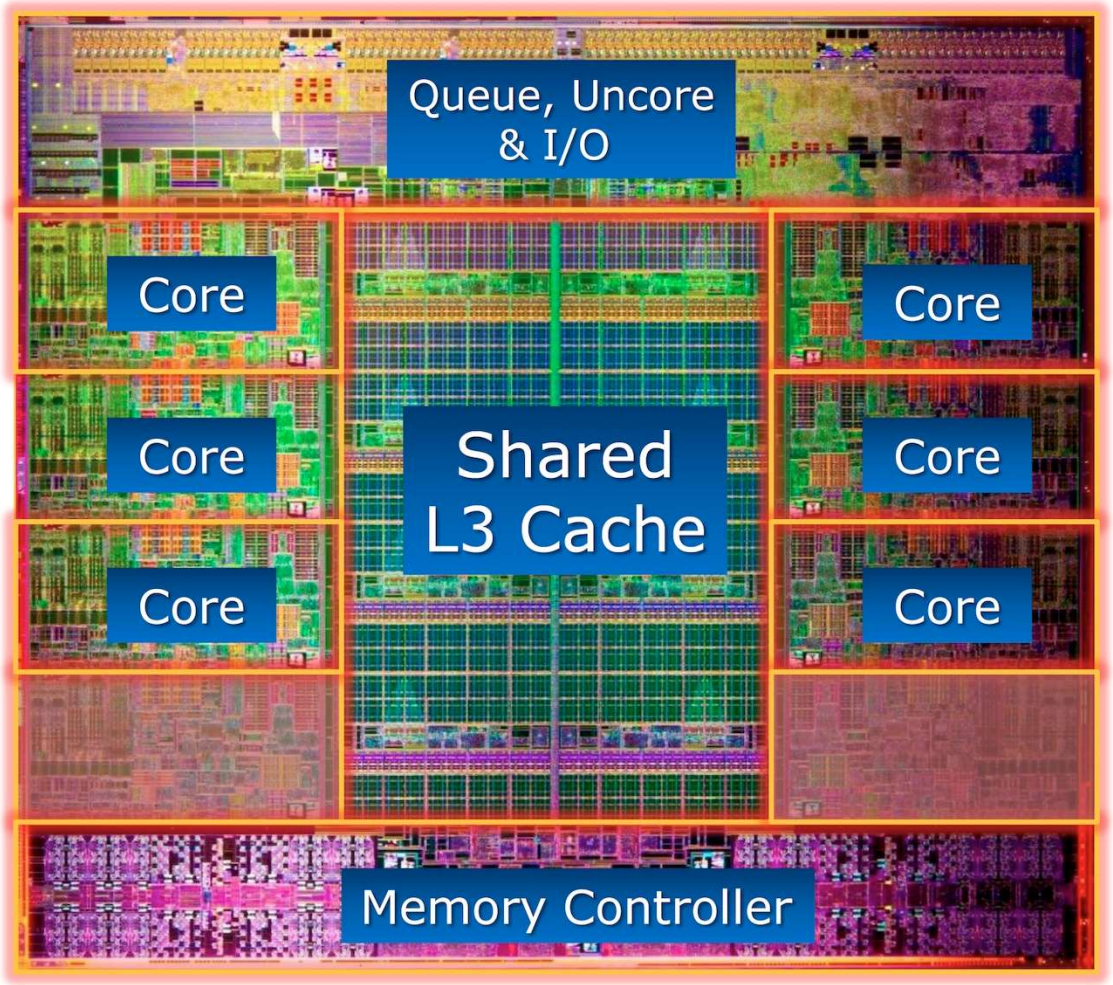
www.intel.com/



hcs.ucla.edu/cluster1.htm

Intel Core i7-3960X (2012) – High-End-PCs

- 2270 million transistors
- 32 nm process
- 3300 to 3900 MHz
- 6 cores / 12 threads
- 435 mm²
- 2011 pins
- 130 watts



Technology development

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price-performance vs. UNIVAC	Adjusted price (2003 $)	Adjusted price-performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$6,107,600	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$4,792,300	318
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$75,390	13,135
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$10,756,800	51,604
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$5,461	154,673
1991	HP 9000/ model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$9,401	16,122,356
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$4,945	239,078,908
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$1,600	11,452,000,000

Number of transistors

Processor	Year of introduction	Transistors
4004	1971	2,250
8008	1972	2,500
8080	1974	5,000
8086	1978	29,000
286	1982	120,000
386	1985	275,000
486 DX	1989	1,180,000
Pentium	1993	3,100,000
Pentium II	1997	7,500,000
Pentium III	1999	24,000,000
Pentium 4	2000	42,000,000

www.intel.com/research/silicon/mooreslaw.htm

Moore‘s Law

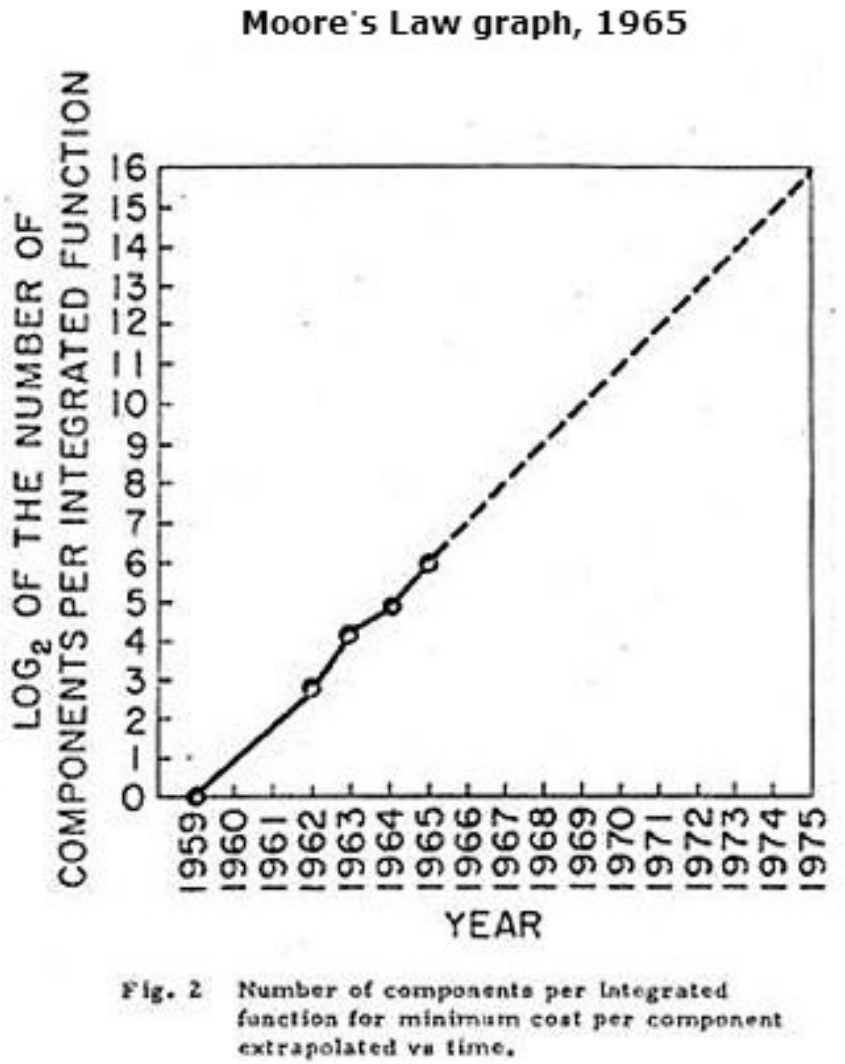


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

Gordon E. Moore

* 03.01.1929, San Francisco

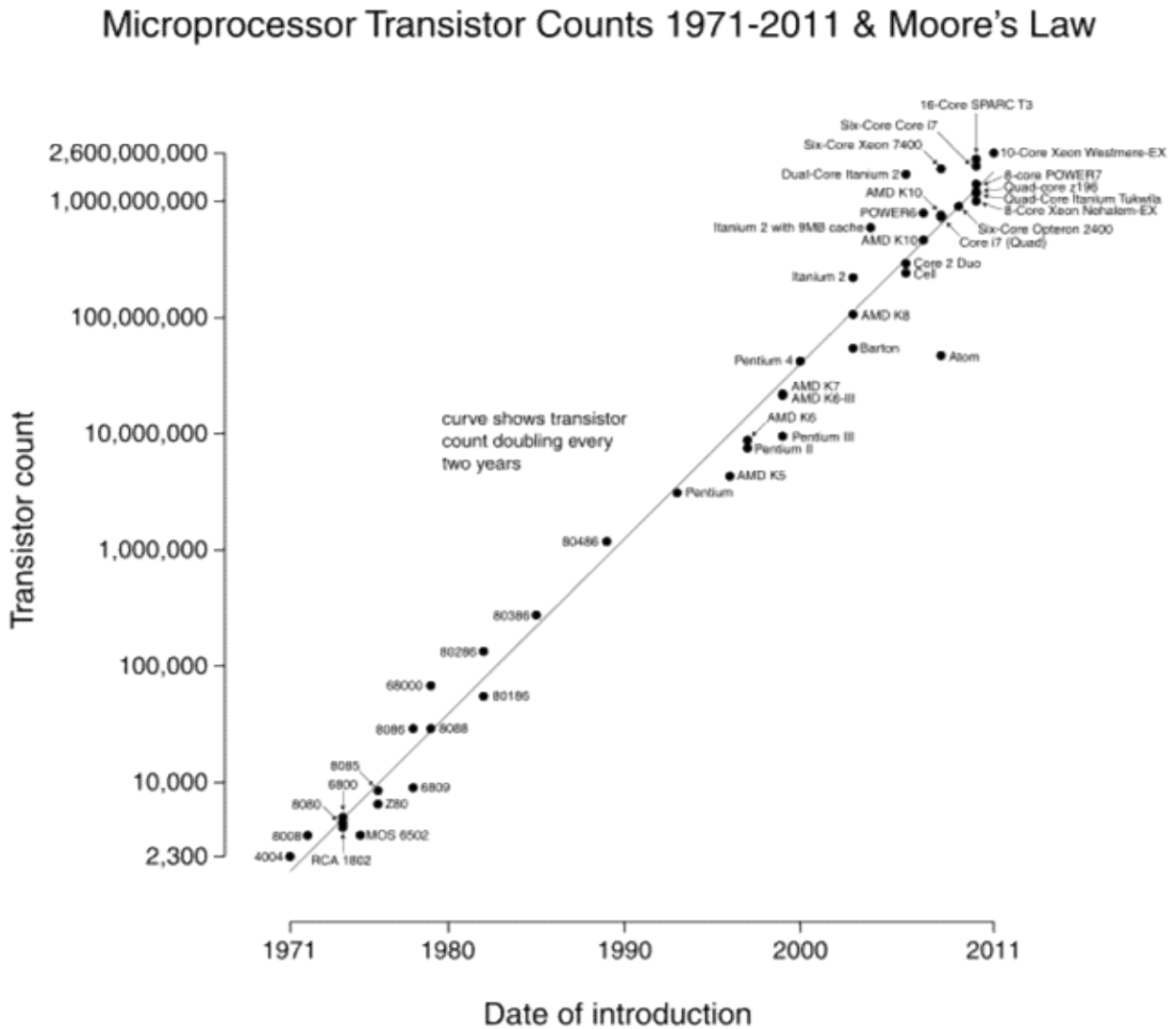
1954 Ph.D. in Chemistry & Physics
Univ. California, Berkeley

1968 founding member Intel Corp.
(Executive Vice President)

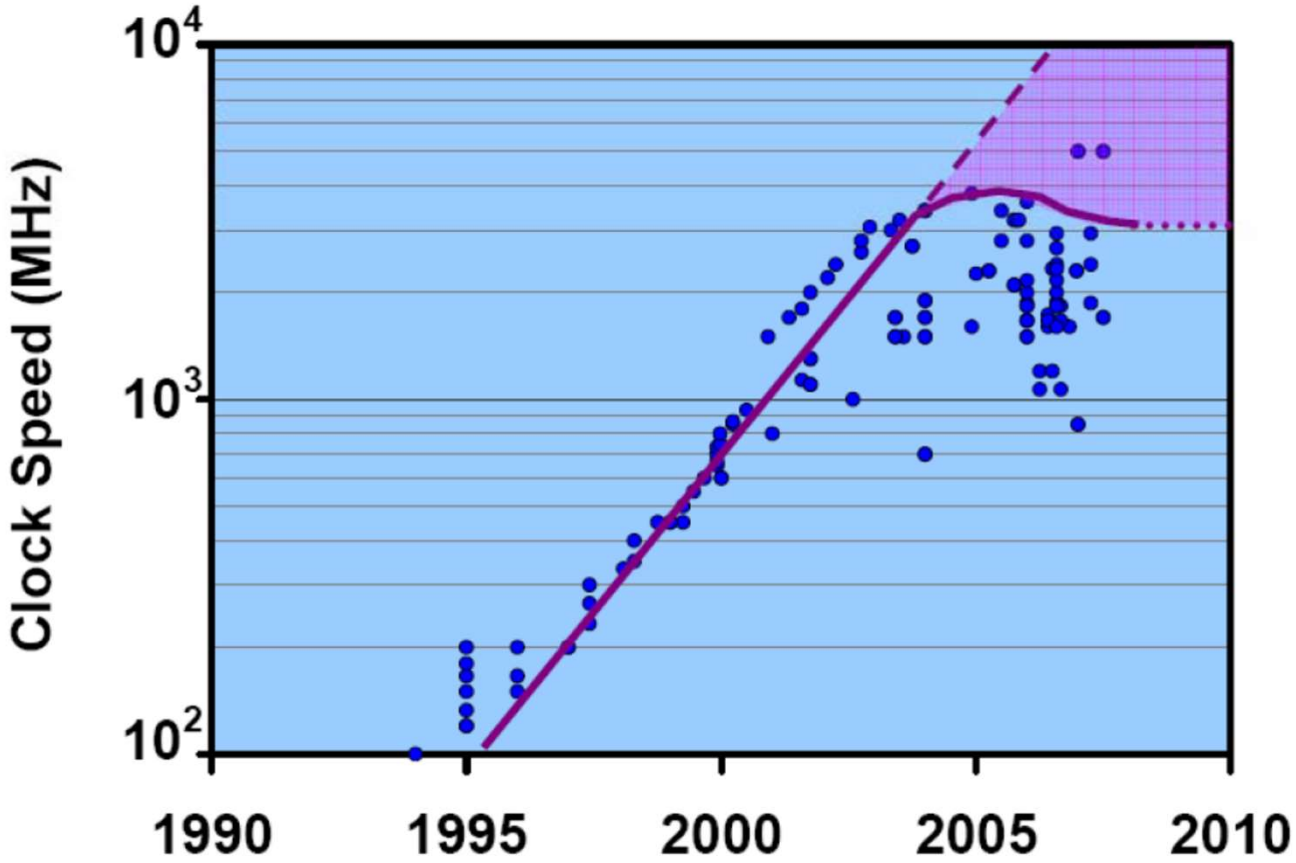
1987 retirement

1997 Chairman Emeritus

Moore‘s Law


Microprocessor Transistor Counts 1971-2011 & Moore's Law
Transistor count
2,600,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
2,300
curve shows transistor count doubling every two years
16-Core SPARC T3
Six-Core Core i7
Six-Core Xeon 7400
10-Core Xeon Westmere-EX
Dual-Core Itanium 2
8-core POWER7
AMD K10
Quad-core z196
Quad-Core Itanium Tukwila
POWER6
8-Core Xeon Nehalem-EX
Itanium 2 with 9MB cache
Six-Core Opteron 2400
AMD K10
Core i7 (Quad)
Core 2 Duo
Cell
Itanium 2
AMD K8
Barton
Atom
Pentium 4
AMD K7
AMD K6-III
AMD K6
Pentium III
Pentium II
AMD K5
Pentium
80486
80386
80286
68000
80186
8086
8088
8085
6800
6809
8080
Z80
8008
MOS 6502
4004
RCA 1802
1971
1980
1990
2000
2011
Date of introduction

Clock speed



Power consumption

When maintaining the computer architecture, it applies:

- computing power ~ clock rate

In CMOS technology, it holds:

- power ~ load capacity · supply voltage² · clock rate
  - clock rate should be increased in new generations
  - voltage is already small (5 V → 1.5 V in the last years)
  - load capacity (number of transistors) is constant or increasing
  - Power ↑↑
  - Problem: heat dissipation

Change in strategy

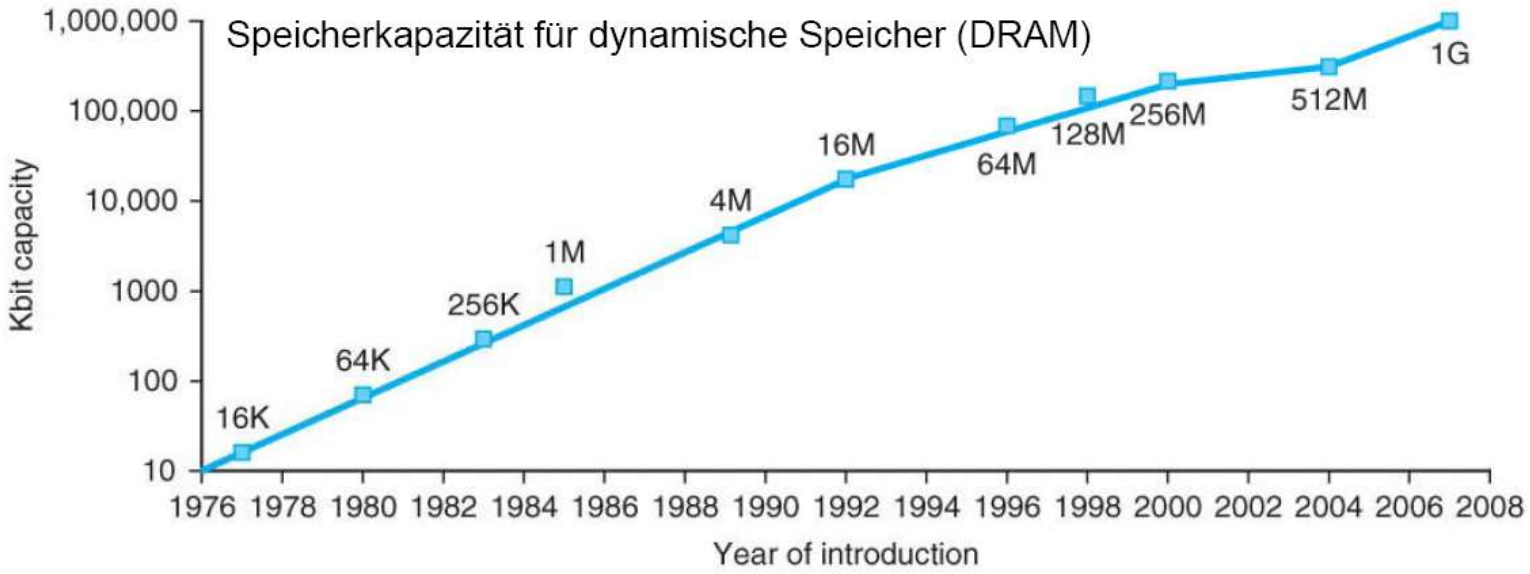
Change of the design strategy

- Lower clock speeds, but several processor cores
- According to Moore's Law, the required footprint of an IC is supposed to shrinking by half every 12 - 24 months
  - Aim: doubling the number of cores every 2 years

Product	AMD Opteron X4 (Barcelona)	Intel Nehalem	IBM Power 6	Sun Ultra SPARC T2 (Niagara 2)
Cores per chip	4	4	2	8
Clock rate	2.5 GHz	~ 2.5 GHz ?	4.7 GHz	1.4 GHz
Microprocessor power	120 W	~ 100 W ?	~ 100 W ?	94 W

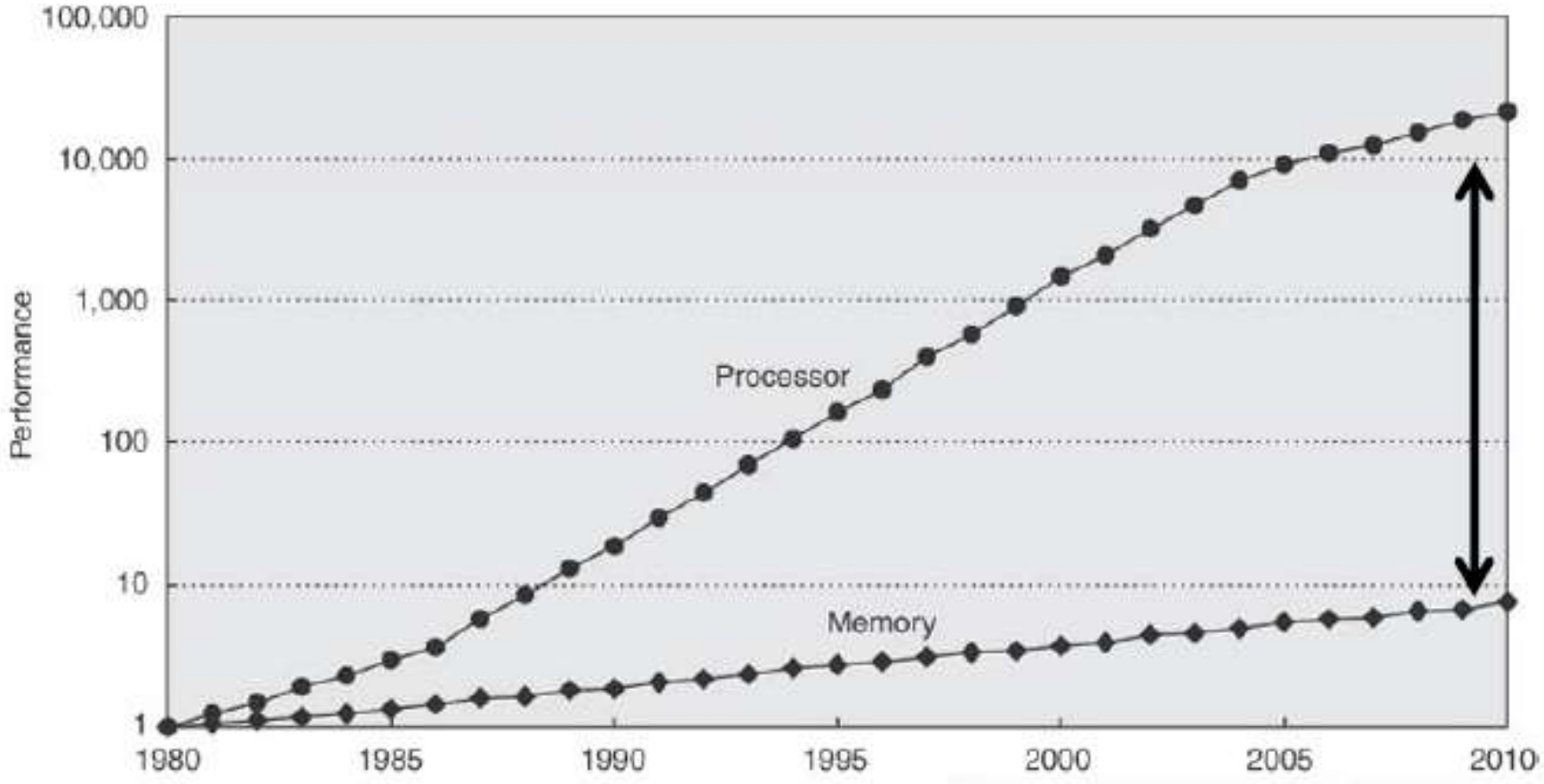
- Problems: Parallel programming, load distribution, communication between the cores, synchronization, …

Storage capacity

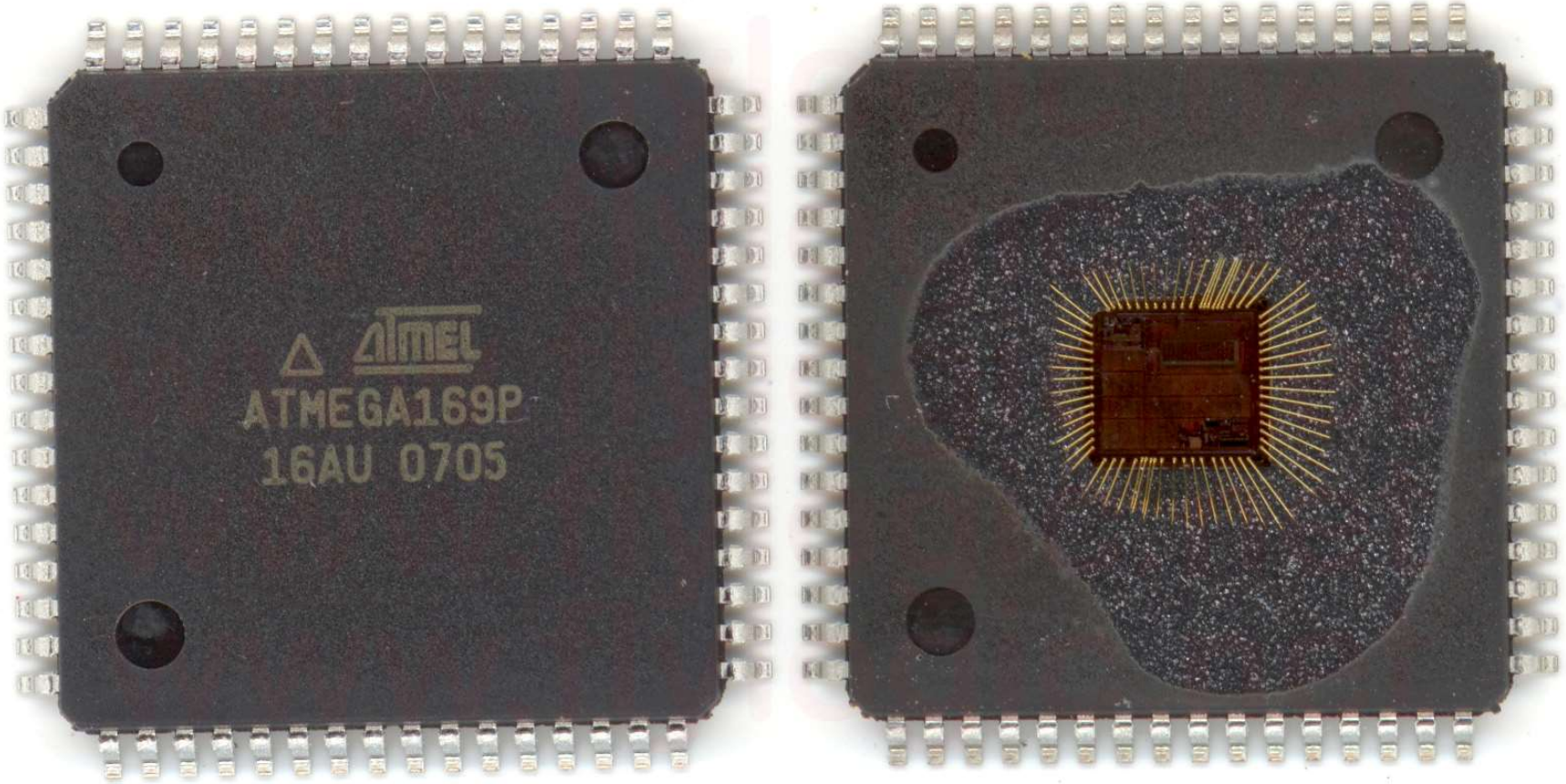


- Storage density increased by 60 % per year; only 30 % of increase in the last years
- access times decreases by approximately 30 % in 5 years

Processor-memory gap

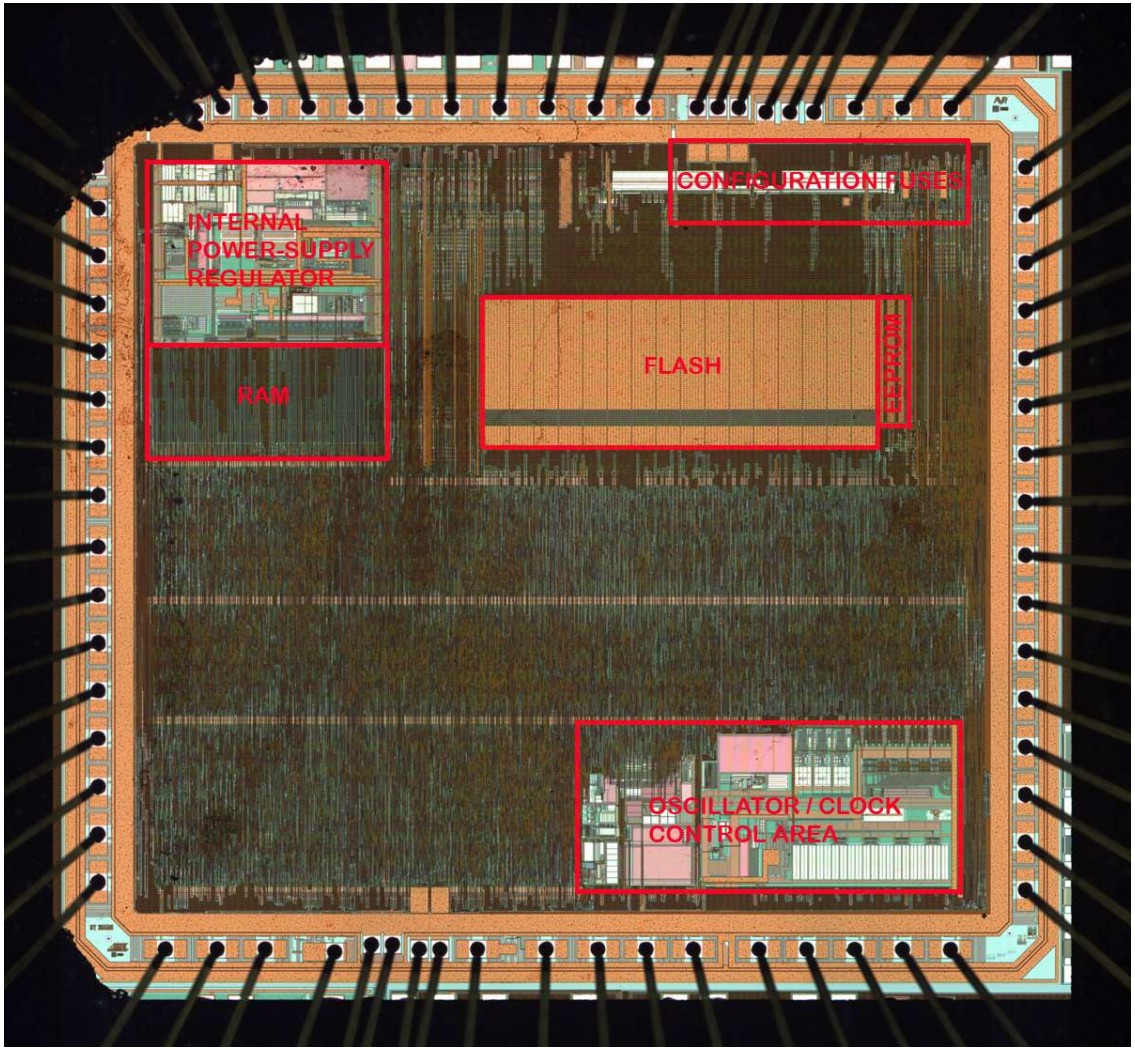


Here it is: The microcontroller!



http://blog.ioactive.com/2007/11/atmega169p-quick-peek.html

Here it is: The microcontroller!



http://blog.ioactive.com/2007/11/atmega169p-quick-peek.html

State of the lecture

- ✓ Introduction

- Chapter 0: basic knowledge
  - Number systems, digital logic, …

- Chapter 1: calculation and control system
  - Basic operations in CMOS, registers, memory, concepts, …

- Chapter 2: Basic peripherals
  - GPIO, ADC, …

…

IMTEK

0. Basic knowledge

Digital signals, character and number representations, digital logic

UNI FREIBURG

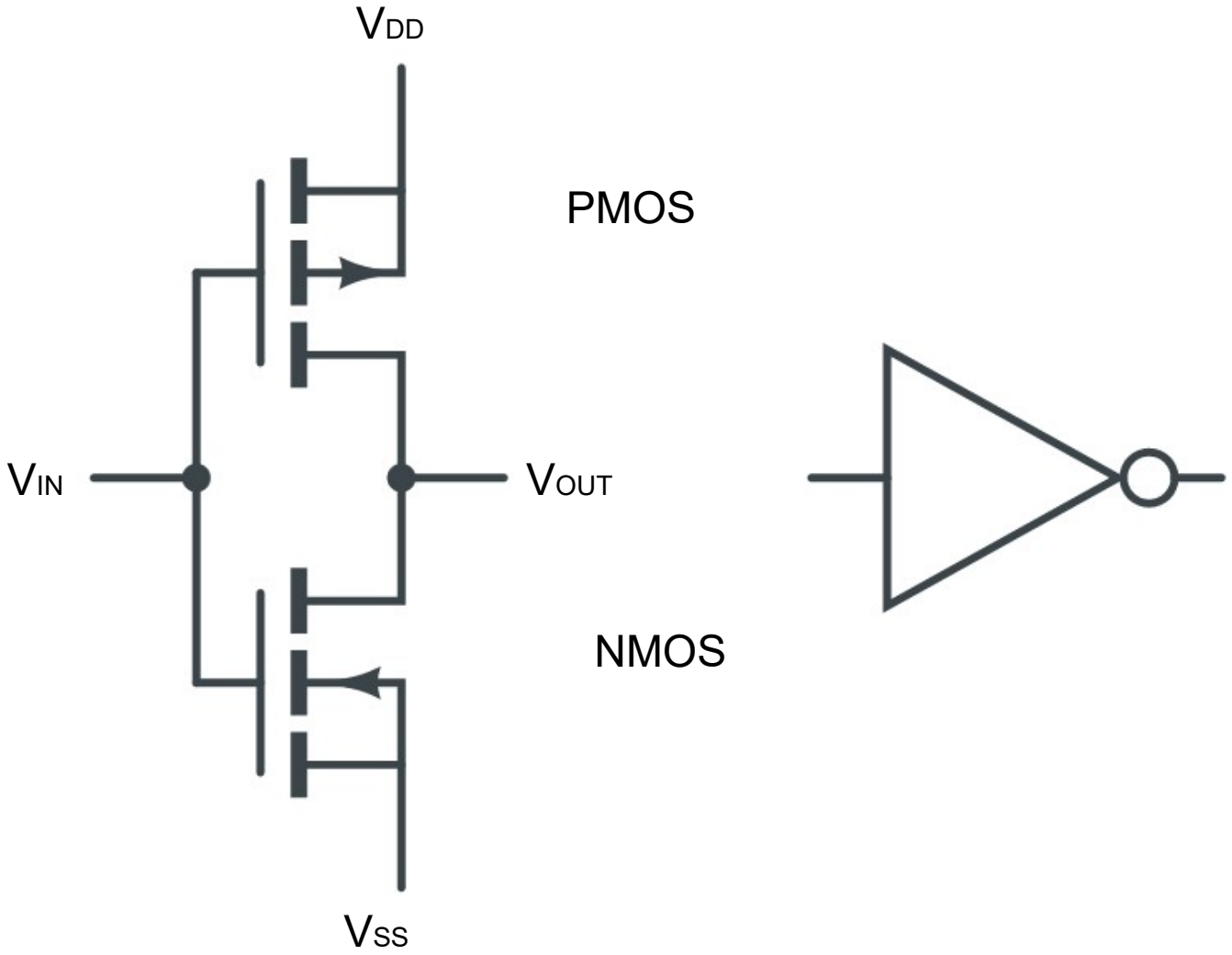
0. Basic knowledge

INTRODUCTION - DIGITAL SIGNALS

High, Low, Z, X

- High signal
  - Connection to the (supply) voltage
  - 24 V, 12 V, 5 V, 3.3 V, 1.8 V, … (generally reffered as V_{DD} for CMOS)
  - Logic 1
- Low signal
  - Connection to ground
  - V_{SS} (for CMOS usually GND)
  - Logic 0
- Z
  - High impedance
  - „floating wire“
  - Neither 1 nor 0
- X
  - any signal

CMOS Inverter im Detail



0. Basic knowledge

CHARACTER- & NUMBER ENCODING

It's all about the numbers

- Computers can't process any text
- Everything which is processed must be represented as a number
- Letters are translated to numbers (see, for example, ASCII (American Standard Code for Information Interchange))

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge	039		072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	⁞	209	Ð	242	‗
012	Form Feed	045	-	078	N	111	o	144	É	177	⁞	210	Ê	243	¾
013	Carriage Return	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Delete	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	-- frei --	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	-- frei --	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	-- frei --	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	-- frei --	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032		065	A	098	b	131	â	164	ñ	197	┼	230	µ		

Bildquelle: www.chip.de

Examples of text encoding

- Hällü Wörld
  (ANSI)
- H„ll W”rld
  (OEM 852)
- HĂ¤llĂĽ WĂ¶rld
  (Windows 1250)

- 01001000 11100100 01101100 01101100 11111100 00100000
  01010111 11110110 01110010 01101100 01100100

Number representations

There are three most frequently encountered number representations in the microcomputer area:

- Binary
  - 0, 1, 10, 11, 100, …
- Decimal
  - 0, 1, …, 9, 10, 11, …
- Hexadecimal
  - 0, 1, …, 9, A, … F, 10, 11, …

Positional notation

General representation of integers

- Most number encodings are based on a positional notation
- Integers are represented by:

$$A = a_{B-1}a_{B-2} \ldots a_1 a_0$$

with $a_i = \{0, 1, \ldots, r - 1\}$ and the position's value r^i

- The value of a number is defined by

$$A = \sum_{i=0}^{B-1} a_i r^i$$

- Usual bases are *r* = 2 (binary), *r* = 10 (decimal) and *r* = 16 (hexadecimal)

Positional notation

- 124
  - 0111 1100 (sometimes written as: 0111 1100b or 0111 1100_2)
  - 0x7C [spoken „hex(adecimal) seven C“; alternativly: 7Ch or $7C_{16}$)
- 255
  - 1111 1111
  - 0xFF
- 256
  - 0001 0000 0000
  - 0x100 (spoken „hexadecimal one-zero-zero“)
  - If limited to 8 bits: 0000 0000 (0x00)

Basic arithmetic operations in binary system

Identical to basic arithmetic operations in the decimal system:

addition

```
    0001
+ 0110
= 0111
```

subtraction

```
    0111
- 0110
= 0001
```

Identical with long numbers

```
    010000000
- 001111111
= 000000001
```

multiplication

```
0101 · 0011
     0000
      0000
       0101
        0101
=     0001111
```

attention:

```
    1111
+   0001
= 10000
```

For a fixed word width of 4 bits
the result is a zero!

Negative numbers

- 5 = 0101
- 127 = 0111 111

- And -127?
- computer don't know a negative sign on the bit level

→ encoding is required

Number encoding: magnitude and sign

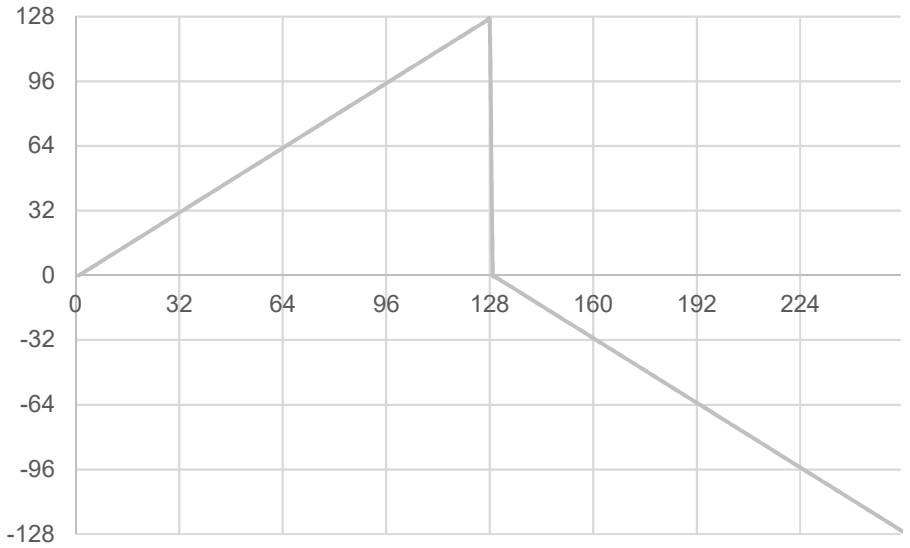
- The MSB (a_{B-1}) determines the sign:

$$A = \begin{cases} \sum_{i=0}^{B-2} a_i r^i & \text{for } a_{B-1} = 0 \\ -\sum_{i=0}^{B-2} a_i r^i & \text{for } a_{B-1} = 1 \end{cases}$$

- The range is $-(2^{B-1}-1) \cdots 2^{B-1} - 1$

Number encoding: magnitude and sign

- Most significant bit indicates sign, the magnitude is remaining
- Example
  - 4 = 0000 0100
  - -4 = 1000 0100
  - 127 = 0111 1111
  - -127 = 1111 1111
  - 0000 0100 + 0000 0001 (4 + 1)
    = 0000 0101 (5)



Interpreted value (Y) over the not interpreted value (X) for 8 bits

- Two issues:
  - 0000 0100 + 1000 0100 = ?
    (1000 1000 would be -8, which is completely wrong …)
  - 1000 0000 ≙ 0000 0000

Number encoding: one's complement

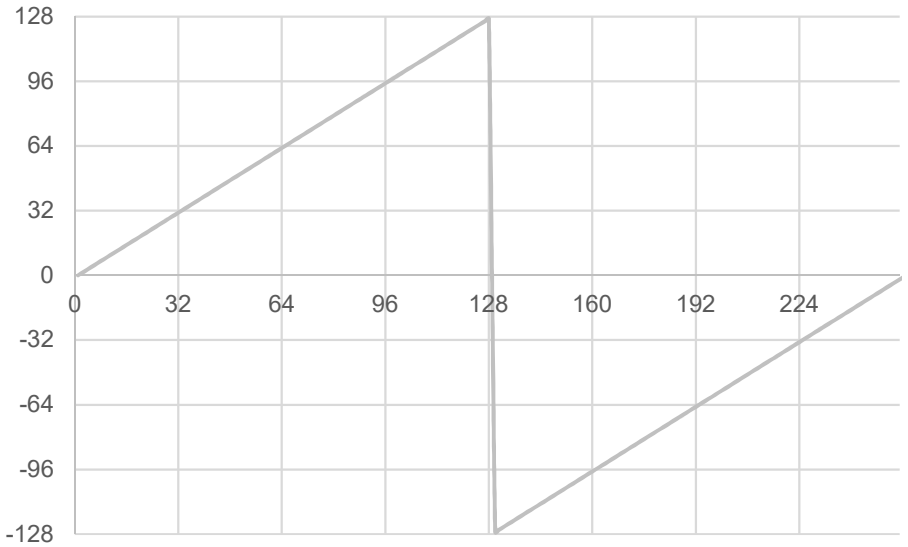
- For negative numbers, the complement is formed (by inverting).
- The MSB (a_{B-1}) is determined to be the sign :

$$A = \begin{cases} \sum_{i=0}^{B-2} a_i r^i & \text{für } a_{B-1} = 0 \\ 2^{B-1} - 1 - \sum_{i=0}^{B-2} a_i r^i & \text{für } a_{B-1} = 1 \end{cases}$$

- The range is $-(2^{B-1} - 1) \cdots 2^{B-1} - 1$

Number encoding: one's complement

- Most significant bit is the sign.
- If the number is negative, the magnitude is inverted
- Example
  - 4 = 0000 0100
  - -4 = 1111 1011
  - 127 = 0111 1111
  - -127 = 1000 0000
  - 0000 0100 + 1111 1011 (4 + (-4))
    = 1111 1111 (0)



Interpreted value (Y) over the not interpreted value (X) for 8 bits

- Again two problems
  - 0000 0110 + 1111 1010 (6 + (-5))= ?
  - 1111 1111 ≙ 0000 0000

Number enoding: two's complement

- For negative numbers, the 2's complement is formed by inverting and adding with one.
- The MSB (a_{B-1}) still determines the sign :

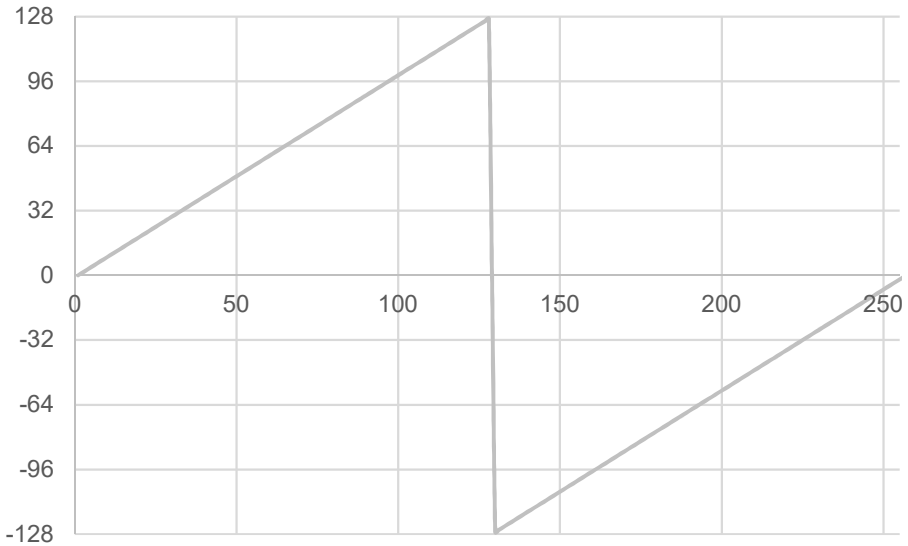
$$A = \begin{cases} \sum_{i=0}^{B-2} a_i r^i & \text{für } a_{B-1} = 0 \\ 2^{B-1} - \sum_{i=0}^{B-2} a_i r^i & \text{für } a_{B-1} = 1 \end{cases}$$

- The range is $-2^{B-1} \cdots 2^{B-1} - 1$

Number Coding: two's complement

- Like one's complement - but negative numbers are incremented by one.
- Example
  - 4 = 0000 0100
  - -4 = 1111 1100
  - 127 = 0111 1111
  - -127 = 1000 0001
  - 0 = 0000 0000
  - -128 = 1000 0000

  - 0000 0110 + 1111 1011 (6 + (-5))= 0000 0001



Interpreted value (Y) over the not interpreted value (X) for 8 bits

Binary arithmetic

	M&S (magnitude & sign)	One‘s (one‘s complement)	Two‘s (two‘s complement)
0001 + 1000			
1010 + 0111			
0110 + 0111			
0110 - 0111			
1010 - 0111			
1111 - 1000			

Binary arithmetic

	M&S		One‘s		Two‘s	
0001 + 1000	0001	(1 + (-0))	1001	(1 + (-7))	1001	(1 + (-8))
1010 + 0111						
0110 + 0111						
0110 - 0111						
1010 - 0111						
1111 - 1000						

Binary arithmetic

	M&S		One‘s		Two‘s	
0001 + 1000	0001	(1 + (-0))	1001	(1 + (-7))	1001	(1 + (-8))
1010 + 0111	0101	(-2 + 7)	0010	(-5 + 7)	0001	(-6 + 7)
0110 + 0111						
0110 - 0111						
1010 - 0111						
1111 - 1000						

Binary arithmetic

	M&S		One‘s		Two‘s	
0001 + 1000	0001	(1 + (-0))	1001	(1 + (-7))	1001	(1 + (-8))
1010 + 0111	0101	(-2 + 7)	0010	(-5 + 7)	0001	(-6 + 7)
0110 + 0111	01101	(6 + 7)	01101	(6 + 7)	01101	(6 + 7)
0110 - 0111						
1010 - 0111						
1111 - 1000						

Binary arithmetic

	M&S		One‘s		Two‘s	
0001 + 1000	0001	(1 + (-0))	1001	(1 + (-7))	1001	(1 + (-8))
1010 + 0111	0101	(-2 + 7)	0010	(-5 + 7)	0001	(-6 + 7)
0110 + 0111	01101	(6 + 7)	01101	(6 + 7)	01101	(6 + 7)
0110 - 0111	1001	(6 - 7)	1110	(6 - 7)	1111	(6 - 7)
1010 - 0111	11001	(-2 - 7)	10011	(-5 - 7)	10011	(-6 - 7)
1111 - 1000	1111	(-7 - 0)	0111	(0 - (-7))	0111	(-1 - (-8))