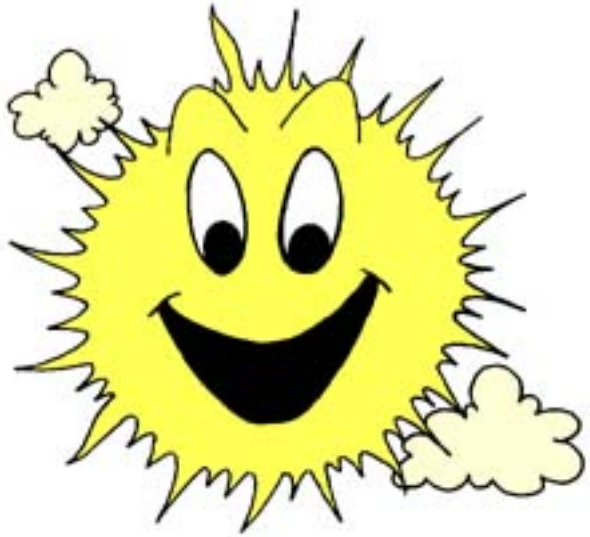
Wiederholung der 1. Vorlesung



- Gesetz von Moore
- Turing-Maschine: universelles Automatenmodell zum Studium algorithmischer Problemstellungen
- Geschichte der Rechnerentwicklung.

→ → → → → → → → → → → → → → → → →

1.3. Historie Software

- Betriebssysteme
  - **interaktiv**er Dialog mit Benutzer ←→ **Batch** (= Stapelverarbeitung)
  - Verwaltung des Plattenspeichers
  - Verwaltung des Arbeitsspeichers
  - Ansteuerung von Drucker, Bildschirm, Peripherie
  - Verwaltung mehrerer parallel arbeitender Benutzer Multiuser
  - „Gleichzeitiges“ Ausführen mehrerer Programme Multitasking

1.3. Historie Software

- Anfang 50er Jahre: IBM Großrechner, überwiegend Batch-Betrieb
- später TSO (time sharing operating system) Mehrbenutzerbetrieb interaktiv
- Ende 60er Echtzeitbetriebssysteme auf PDP, VAX

Programme nur auf *einer* HW lauffähig

- 1969 Portierung eines Spiels bei AT&T, Bell
- 1969 Kenneth Thompson schreibt UNIX, entwirft Sprache ‘B‘ portiert auf beides auf PDP 7.
- Dennis Ritchie formt aus ‘B‘ die Sprache C.
- 1973 Ritchie schreibt UNIX in C um → portabel
- 1975 UNIX etabliert, Weiterentwicklung, viele Derivate!!

1.3. Historie Software (2)

- Mitte der 80er
  - Vernetzung, NFS (Network File System)
  - Graphik, Oberfläche X entsteht
  - UNIX auf Minicomputern und Workstations führend
- 1981 MS-DOS. Microsoft programmiert für IBM
  - Nichtexclusiver Lizenzvertrag mit IBM für MSDOS
  - Microsoft besitzt Verkaufslizenz für Basic-Interpreter
  - MS Windows: ab Version 3.1 sehr erfolgreich
- Free Software Foundation FSF: Freiheit für Software
  - 1985 Richard Stallmann GNU-Projekt (GNU is not Unix)
  - Vorhandene Pakete werden nachprogrammiert
    GCC C-Compiler, emacs, Koordination durch Internet
- 1991 Linus Torwald: Unix für PC‘s: LINUX

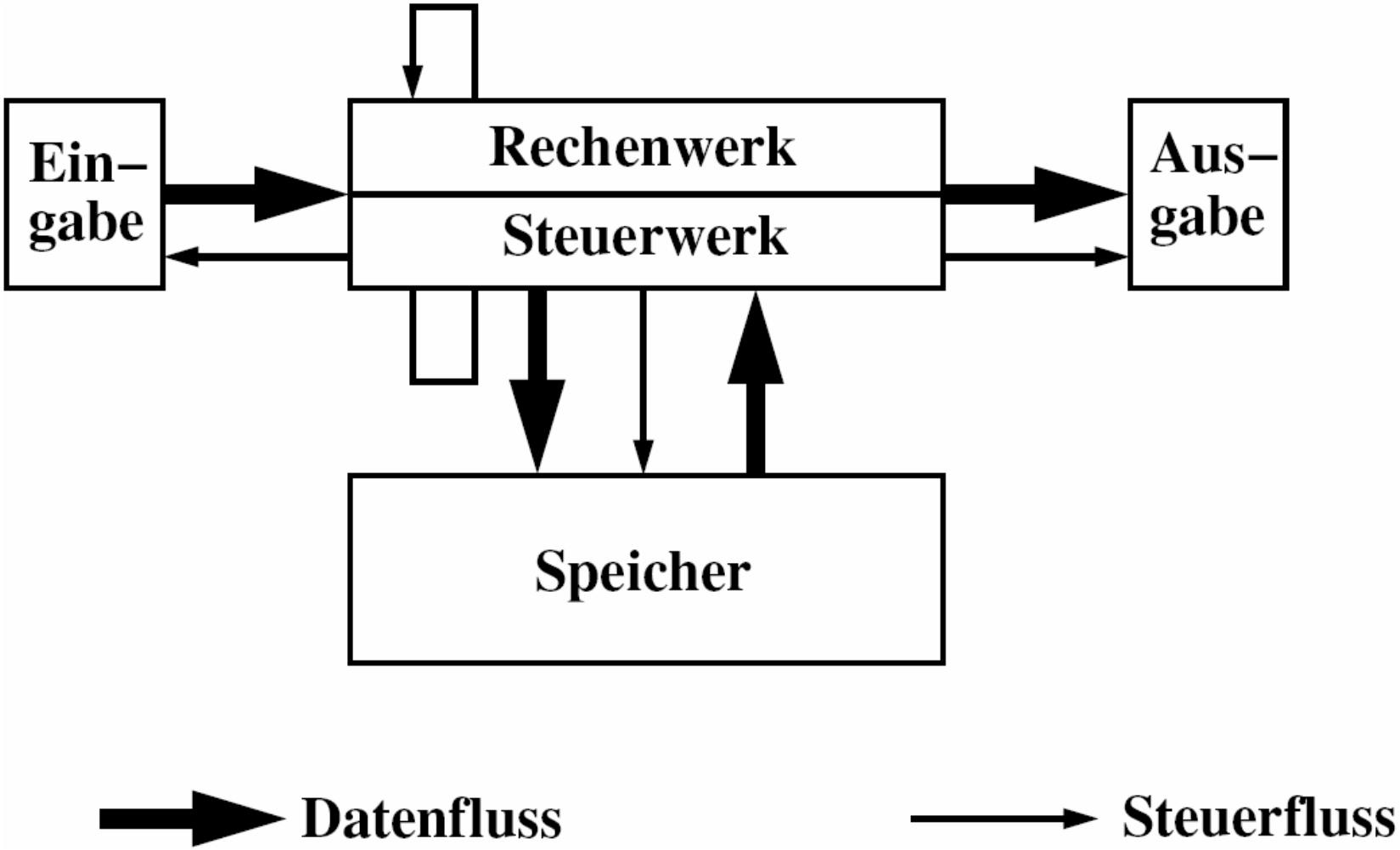
1.3. Historie der Netze

- 1967: Entwicklung des ARPANET beginnt
- 1972: Vernetzung von 40 Rechnern im ARPANET, email wird beliebt
- 1976: Unix to Unix Copy Protokoll (UUCP) bei Bell Labs entwickelt
- 1979: Computer Science Research Network (CSNET)
- 1979: Unix User Network (USENET), Nachrichtenartikel werden zwischen Rechnern kopiert und weltweit Anwendern verfügbar gemacht

1.4 Komponenten eines Rechners

- Peripherie (Tastatur, Monitor etc.): Eingabe, Ausgabe
- Speicher (RAM, HDD): Speicherung
- Rechnerkern (Prozessor), der datenverarbeitende Teil wird als Rechenwerk bezeichnet, der steuernde Teil wird als Steuerwerk bezeichnet

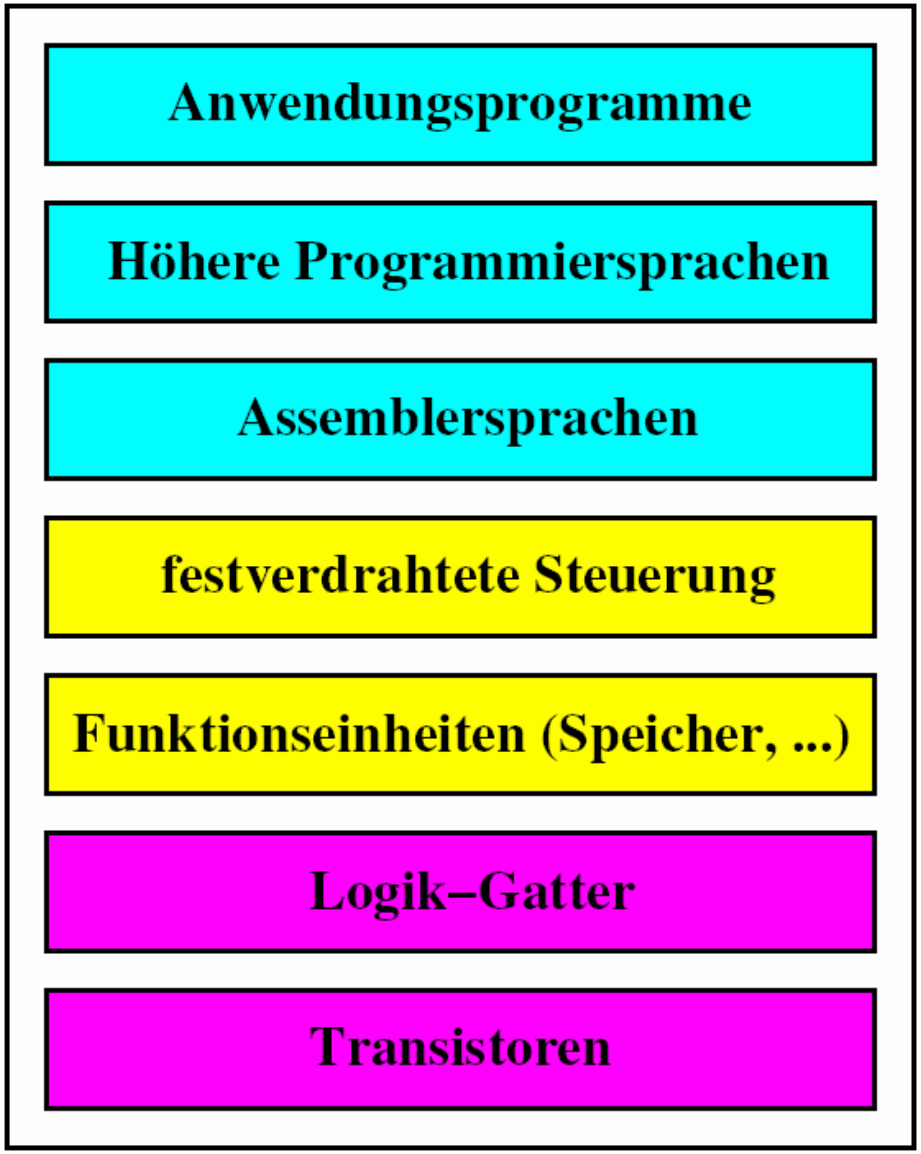
1.4 Daten und Steuerfluss



1.5 Programmierbare Rechner

- moderne Rechner sind programmierbar
- für die Steuerung wird ein Algorithmus verwendet (unabhängig von Rechnerstruktur)
- die Folge von Anweisungen wird vom Rechenwerk ausgeführt
- Algorithmus liegt im Hauptspeicher
- Daten liegen im Hauptspeicher
- vom Steuerwerk werden die einzelnen Befehle interpretiert und die Anweisungen an das Rechenwerk geleitet

1.6 Schichtenmodell



2. Zahlensysteme

1. Darstellung von Zahlen
2. Positive Zahlen
3. Positive- und negative Zahlen
4. Gebrochene Zahlen
    1. Festkomma
    2. Gleitkomma (siehe 2.8)
5. 1er- und 2er Komplement
6. Addition und Subtraktion
7. Little Endian – Big Endian
8. Fließkomma- und Gleitkommaformate

2.1 Arithmetische Operationen

- Grundlage aller Funktionalität von Rechnern bildet die dahinter liegende Arithmetik
- Zum Verständnis von Rechner gehört das Verständnis der zugrunde liegenden Arithmetik
- Um die Arithmetik von Rechnern zu verstehen muss man wissen wie Zahlen im Rechner dargestellt werden.

- Welche Darstellung ist für Berechnungen in Maschinen besonders vorteilhaft?

Informatik

Ergebnis Multiplikation
von 136 und 14?

136 * 14 = 1904

136 * 14 = 1838

$136_{(10)} * 14_{(10)} = 1904_{(10)}$

$136_{(16)} * 14_{(16)} = 1838_{(16)}$

- Für die Durchführung arithmetischer Operationen muss die verwendete Basis bekannt sein
- Weitere Konventionen der Darstellung müssen bekannt sein

2.1 Darstellung zur Basis 2

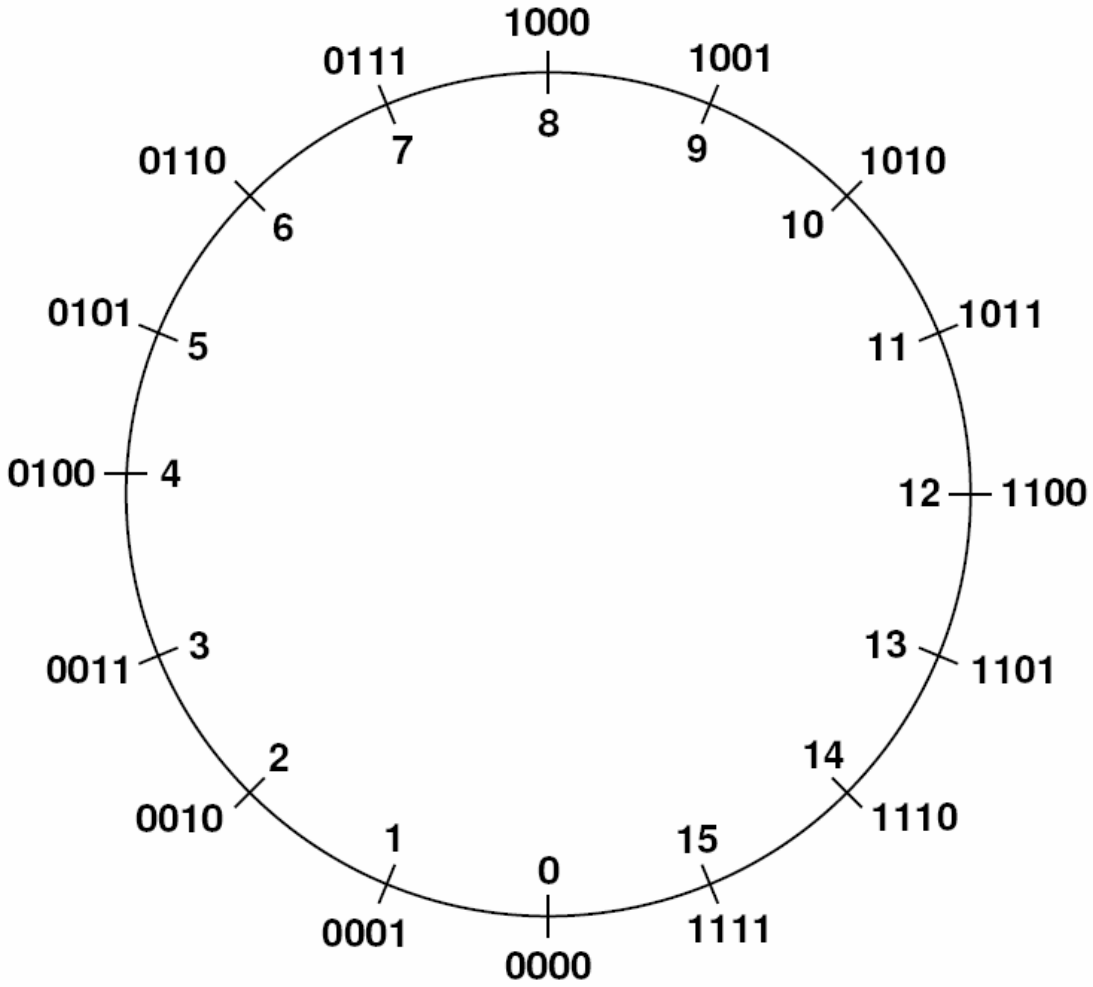
Heutige Rechner arbeiten intern nur mit 0 und 1.

Alle Berechnungen werden zur Basis 2 (Dual-, Binärsystem) durchgeführt.

Elementare Zahlenarten:

- Natürliche Zahlen (positiv, z.B. 1, 2, 3, ...)
- Ganze Zahlen (positiv und negativ, z.B. -5, 0, 2, 42, ...)
- Rationale Zahlen (z.B. -2/3, 7/2, 13/17, ...)
- Reelle Zahlen (ganze und rationale sowie irrationale Zahlen wie π oder $\sqrt{2}$)

2.2 Integer Zahlen (ohne Vorzeichen)



$$Z = \sum_i x_i Y^i,$$

$= x_3*2^3 + x_2*2^2 + x_1*2^1 + x_0*2^0$

$= x_3*8 + x_2*4 + x_1*2 + x_0*1$

Wertebereich:

$0 \leq Z \leq 2^n - 1$

2.2 Beispiel Umrechnung in Basis 2

$(235)_{10} = (z)_2$

235 : 2 = 117,	Rest 1	$2^0 * 1$	1	+
117 : 2 = 58,	Rest 1	$2^1 * 1$	2	+
58 : 2 = 29,	Rest 0	$2^2 * 0$	4	+
29 : 2 = 14,	Rest 1	$2^3 * 1$	8	+
14 : 2 = 7,	Rest 0	$2^4 * 0$		
7 : 2 = 3,	Rest 1	$2^5 * 1$	32	+
3 : 2 = 1,	Rest 1	$2^6 * 1$	64	+
1 : 2 = 0,	Rest 1	$2^7 * 1$	128	

11101011 = $(235)_{10}$

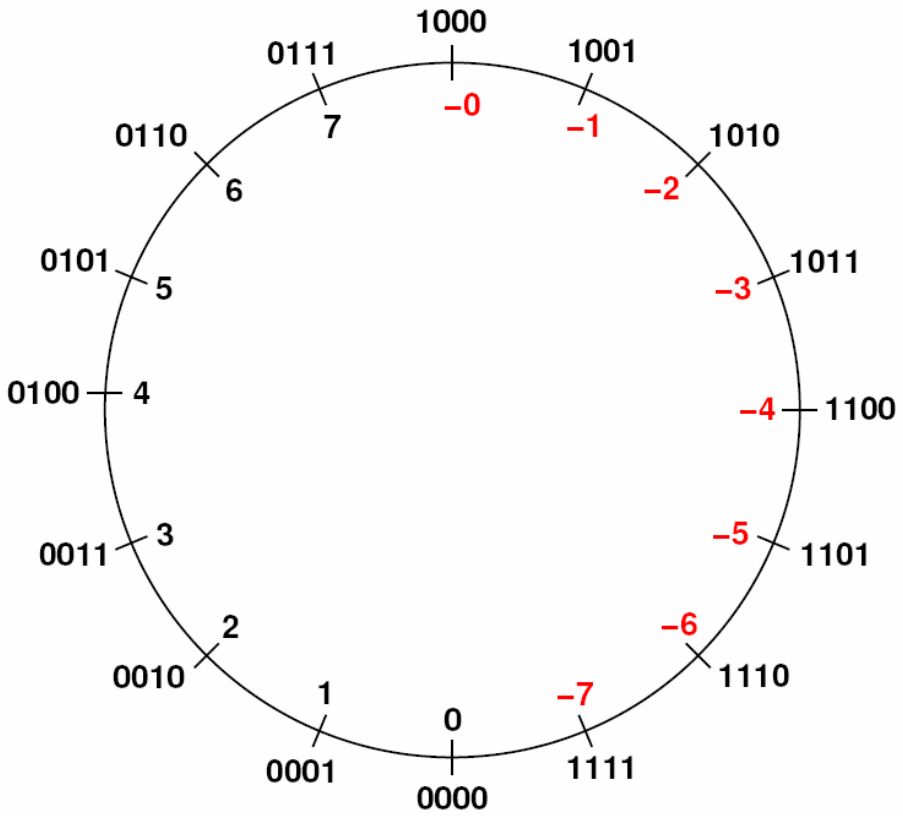
2.2 Beispiel Umrechnung in Basis 8

$(235)_{10} = (z)_8$

235 : 8 = 29,	Rest 3		$8^0 * 3$	3 +
29 : 8 = 3,	Rest 5		$8^1 * 5$	40 +
3 : 8 = 0,	Rest 3		$8^2 * 3$	192 +

$353_8 = (235)_{10}$

2.3 Integer Zahlen (mit Vorzeichen)



$$Z = (-1)^{x_3} * [x_2*2^2 + x_1*2^1 + x_0*2^0]$$
$$= \pm 1 * [x_2*4 + x_1*2 + x_0*1]$$

Wertebereich:

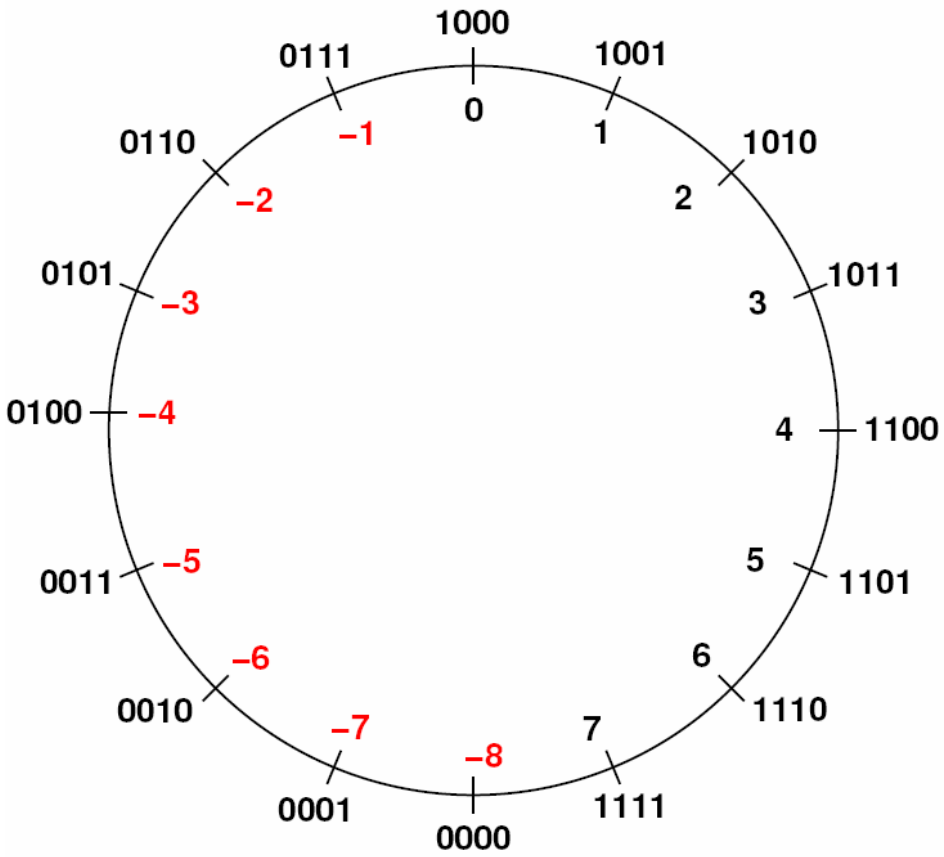
$$-2^{n-1}+1 \leq Z \leq 2^{n-1}-1$$

Most Significat Bit (MSB) definiert das Vorzeichen.

Vorzeichen muss gesondert ausgewertet werden.

Zwei Darstellungen der 0, nämlich +0 und -0

2.4 Integer Zahlen (Binary Offset)



Der kleinst mögliche Wert wird mit 0000... und der größt mögliche Wert wird mit 1111... dargestellt.

Wertebereich:

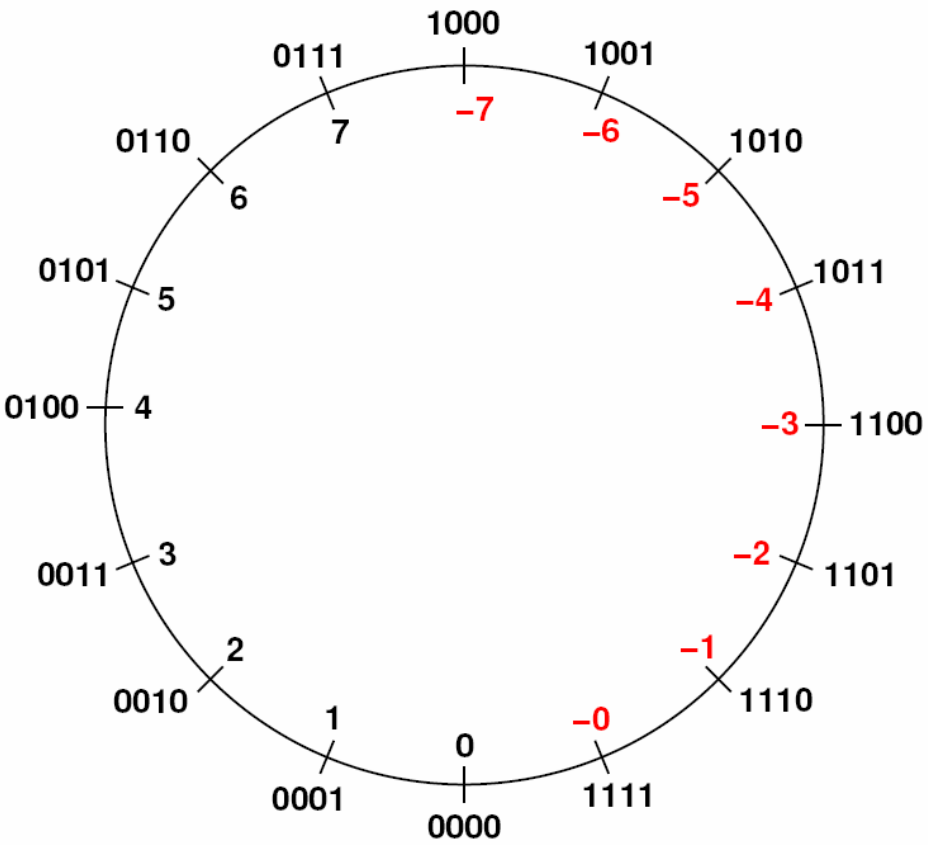
$-2^{n-1} \leq Z \leq 2^{n-1} - 1$

Z = D – O; D: binäre Darstellung, O = 8

Nur eine Darstellung der 0.

Findet Verwendung beim Exponent der IEEE 754

2.5 Integer Zahlen (1er-Komplement)



Most Significant Bit (MSB) definiert das Vorzeichen.

positive Zahl: MSB = 0, (3) niederwertigen Bits werden wie gewohnt ausgewertet

negative Zahl: MSB = 1, (3) niederwertigen Bits werden invertiert und anschließend ausgewertet

Wertebereich:

$$-2^{n-1}+1 \leq Z \leq 2^{n-1}-1$$

2.5 Beispiel 1er-Komplement

-3 soll binär als 1er-Komplement dargestellt werden

- binäre Darstellung des Betrags
- $(3)_{10} = (0011)_2$
- Setzen des Vorzeichens und Komplement der Bits
- $(-3)_{10} = (1100)_{2\text{-}1K}$

2.4 Gebrochen Zahlen

Bisher haben wir die binäre Darstellung der ganzen Zahlen kennen gelernt: Integer

$$Z = \sum_i x_i Y^i, \quad \text{mit } 0 \leq i \leq n,\ Y = 2$$

Darstellung gebrocher Zahlen (Rationaler Zahlen):

- Festkomma-Darstellung
- Fließkomma-Darstellung (Floating Point)

2.4 Festkomma-Zahlen

$$a = \sum_{i=-m}^{i=n} z_i b^i$$

$$= z_n b^n + z_{n-1} b^{n-1} \ldots z_1 b + z_0 + z_{-1} b^{-1} + \ldots z_{-m} b^{-m}$$

n+1 Bits für **Vorkomma**-, **m** Bits für **Nachkomma**stellen

Es ist Konvention, die Ziffern mit absteigendem Index *i* als Liste darzustellen und zwischen z_0 und z_{-1} ein Komma einzufügen.

$$a = z_3 \times b^3 + z_2 \times b^2 + z_1 \times b^1 + z_0 \times b^0 +$$

$$z_{-1} \times b^{-1} + z_{-2} \times b^{-2} + z_{-3} \times b^{-3} + z_{-4} \times b^{-4}$$

2.4 Festkomma-Zahlen (2)

$$(123{,}4567)_{10} = 0\times1000 + 1\times100 + 2\times10 + 3\times1 + \frac{4}{10} + \frac{5}{100} + \frac{6}{1000} + \frac{7}{10000}$$

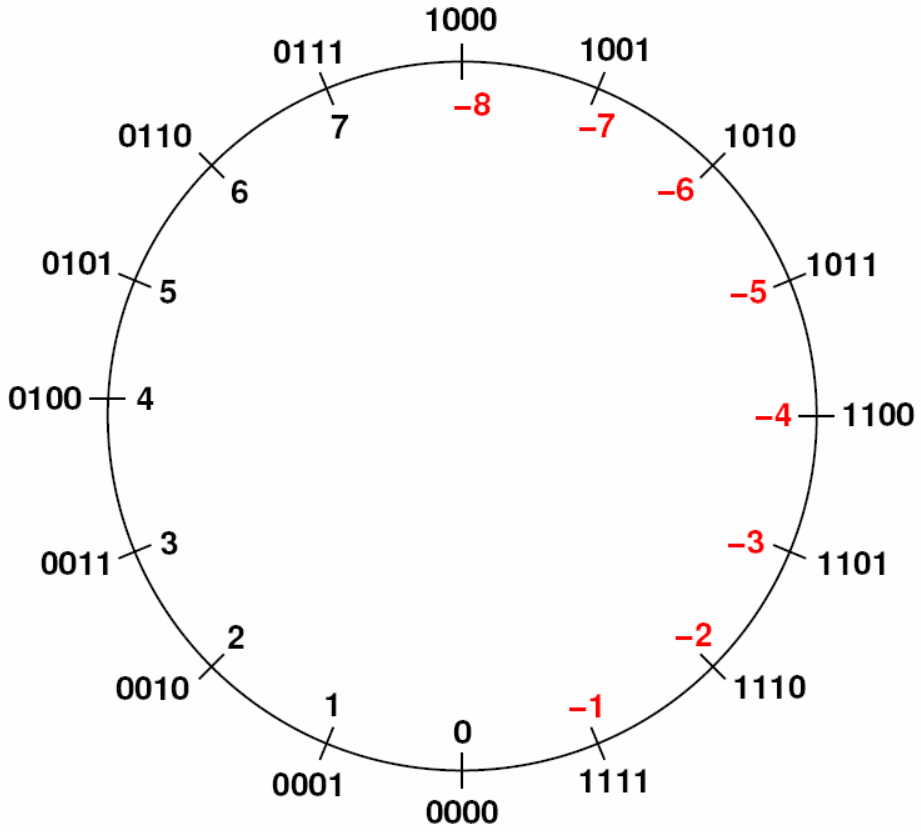
in gleicher Weise kann man binäre Zahlen darstellen:

$$(1001{,}1101)_2 = 1\times8 + 0\times4 + 0\times2 + 1\times1 + \frac{1}{2} + \frac{1}{4} + \frac{0}{8} + \frac{1}{16}$$

In diesem Beispiel ist also $(1001{,}1101)_2 = (9{,}8125)_{10}$

→ Bei Festkomma-Zahlen ist der Wertebereich stark eingeschränkt!

2.5 Integer Zahlen im 2er-Komplement



Most Significant Bit (MSB) definiert das Vorzeichen.

positive Zahl: MSB = 0, (3) niederwertigen Bits werden wie gewohnt ausgewertet

negative Zahl: MSB = 1, 3 niederwertigen Bits werden invertiert, **1 addiert** und anschließend ausgewertet

Wertebereich:

$$-2^{n-1} \leq Z \leq 2^{n-1} - 1$$

2.5 Beispiel 2er-Komplement Festkomma

Informatik

Umwandlung **binär → dezimal**, MSB = 1

$a = (1011,0110)_2$

- falls MSB = 1 → Zahl ist negativ, dann
- Betrag berechnen
  - Komplement aller Bits, 1 addieren
  - → $(1011,0110)_{2\text{-}2K} = (0100,1001)_2 + (1)_2$
    $= (0100,1010)_2$
  - $|a| = (4,625)_{10}$
- Vorzeichen negativ
- **a = -4,625**

2.5 Beisp. 2er-Komplement Festkomma (2)

Informatik

Umwandlung **dezimal → binär**, a < 0

$a = (-4{,}625)_{10}$

- falls a < 0 → Zahl ist negativ, dann
- Betrag berechnen:

Vorkomma	*Nachkomma*
4:2 = 2, R=0	**0,625* 2 = 1, r=0,25**
2:2 = 1, R=0	**0,25 * 2 = 0, r=0,5**
1:2 = 0, R=1	**0,5 * 2 = 1, r=0**

→ |a| = 0100,1010

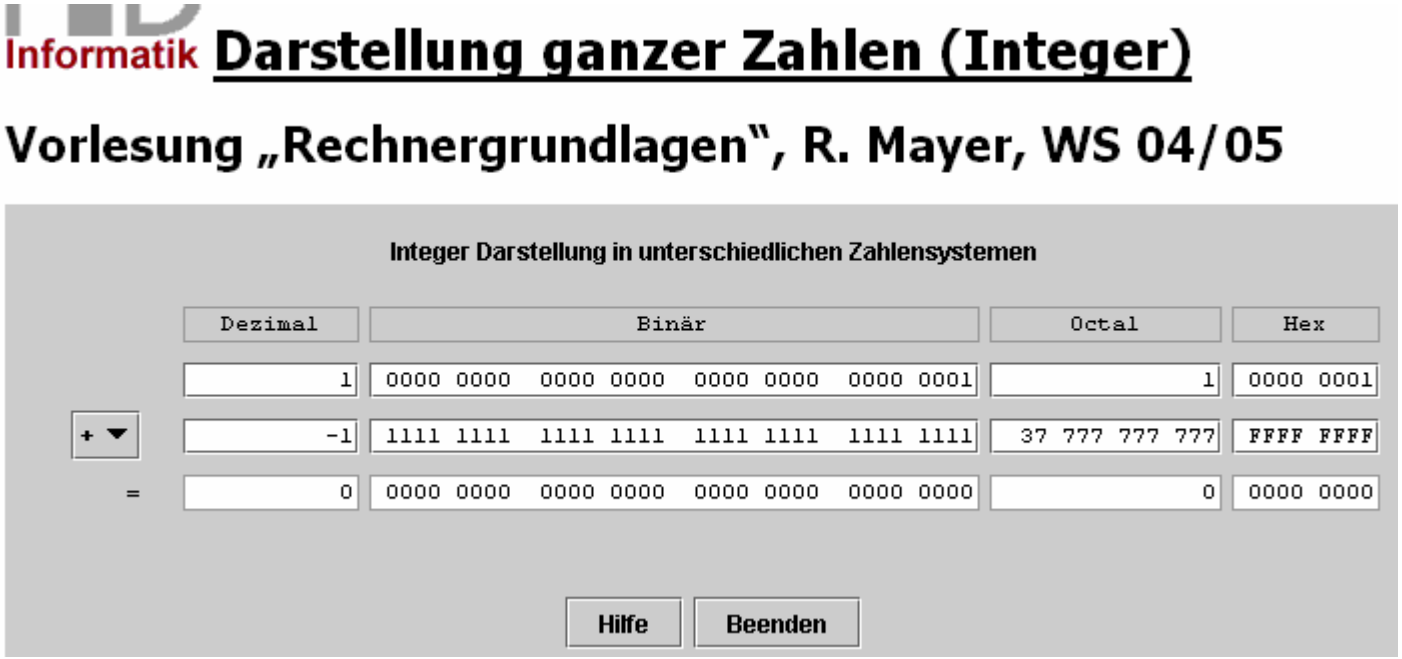
  - Komplement aller Bits, 1 addieren
  - → $(0100{,}1010)_{2\text{-}2K} = (1011{,}0101)_2 + (1)_2$
    $= (1011{,}0110)_2$

- **$a = (1011{,}0110)_2$**

2.5 Beispiel Applet: Integer Darstellung

Beispiel für binäre Darstellung von 32-Bit Integer sowie den vier Grundtrechenarten:

http://www.fbi.fh-darmstadt.de/~rmayer/RG/ra/IntegerDarst.html



2.5 Vorteil der 2er-Komplement-Darstellung

- Es existiert nur eine Null.
  - Null ist zu sich selbst Komplement
  - $(0000)_{2k} = (FFFF)_{1k} + 1 = 0$
- Das Vorzeichenbit muss für Addition und Subtraktion nicht gesondert ausgewertet werden.
- Signifikante Vereinfachung der Berechnung.
- 2er Komplement wird in der Praxis fast ausschließlich eingesetzt.

2.6 Addition und Subtraktion

Addition:

1 + 0 =

0 + 1 = **1**, kein Übertrag (Carry)

1 + 1 = **0**, Übertrag **1**

Subtraktion:

1 - 0 = **1**, kein Übertrag

0 - 1 = **1**, Übertrag **1**

1 - 1 = **0**, kein Übertrag

2.6 Beispiele Addition, Subtraktion

Informatik

Ganze Zahlen ohne Vorzeichen

	6_{10}	0110
+	3_{10}	0011
C		110
=	9_{10}	1001

	6_{10}	0110
-	3_{10}	0011
C		011
=	3_{10}	0011

Ganze Zahlen mit Vorzeichen (2er-Komplement)

	6_{10}	0110
$+(-3_{10})$		1101
C		100
=	3_{10}	10011

	3_{10}	0011
$+(-6_{10})$		1010
C		010
$=-3_{10}$		1101

2.6 Beispiele Addition, Subtraktion (2)

Informatik

Ganze Zahlen ohne Vorzeichen

15_{10}	1111
$+\ 1_{10}$	0001
C	111
$=16_{10}$	10000

- Übertrag (Carry, C, CY)
- Überlauf (Overflow, OV)

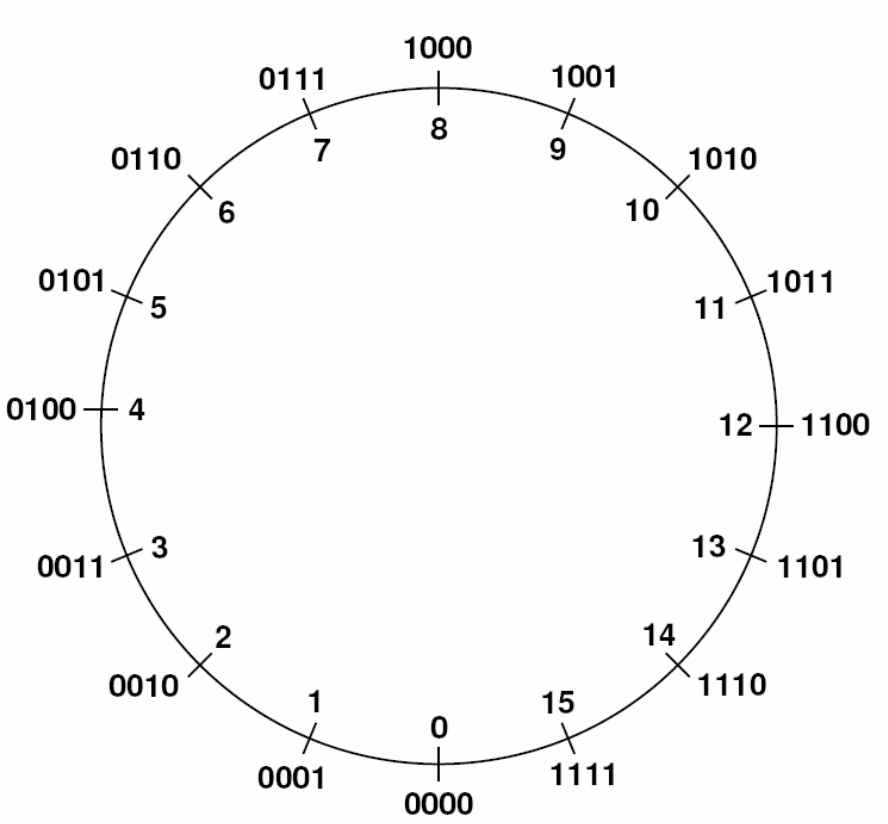
Ganze Zahlen mit Vorzeichen (2er-Komplement)

6_{10}	0110
$+\ 2_{10})$	0010
C	110
$=-8_{10}$	1000

-7_{10}	1001
$+(-6_{10})$	1010
C	000
$=+3_{10}$?	10011

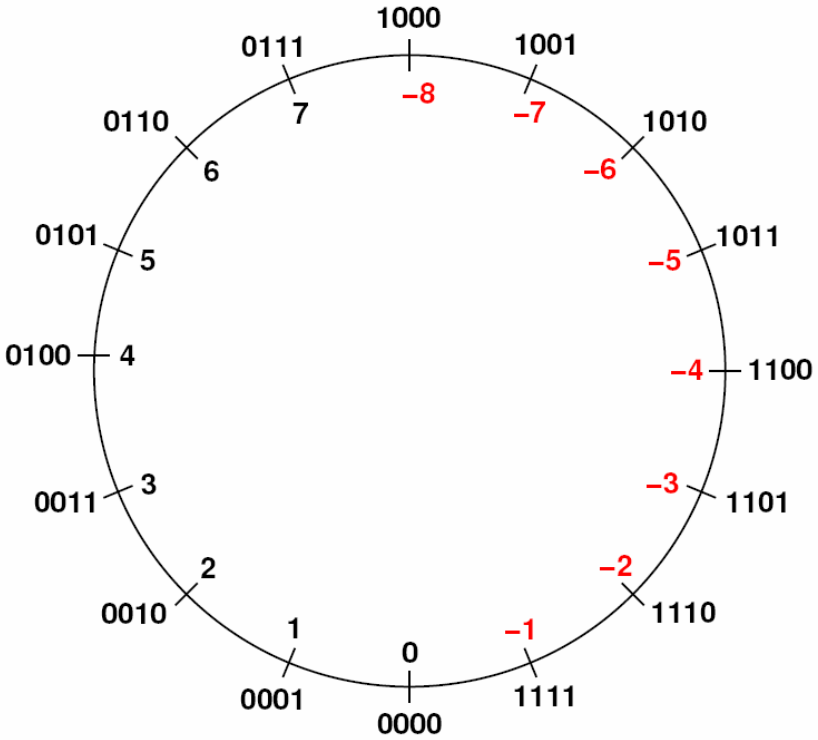
2.6 Übertrag (Carry)

- Ein Übertrag entsteht, wenn bei Operationen mit nur positiven Zahlen der Wertebereich überschritten wird.



2.6 Überlauf (Overflow)

- Ein Überlauf entsteht, wenn bei Operationen von 2er-Komplement-Zahlen der Wertebereich überschritten wird.
- Ein Überlauf liegt dann vor, wenn beide Summanden ein gleiches Vorzeichen haben und das Ergebnis ein anderes.



2.7 Little Endian – Big Endian

Bei der Interpretation, z.B. einer Integer-Zahl, ist zu unterscheiden, ob das niedrigstwertige Byte an der höchsten Adresse („big endian“) oder der niedrigsten Adresse („little endian“) gespeichert wird.

Beispielsweise Speicherung eines 4 Byte Langwort:

MSB → Byte3 Byte2 Byte1 Byte0 ← LSB

Speicheradresse	Little Endian	Big Endian
BasisAddresse+0	Byte0	Byte3
BasisAddresse+1	Byte1	Byte2
BasisAddresse+2	Byte2	Byte1
BasisAddresse+3	Byte3	Byte0

Little Endian: PC (Intel, AMD), DEC, ...

Big Endian: Apple, IBM370

2.7 *Beisp: Fileformate Little / Big Endian*

Informatik

- **Adobe Photoshop** -- Big Endian
- **BMP (Windows and OS/2 Bitmaps)** -- Little Endian
- **DXF (AutoCad)** -- Variable
- **GIF** -- Little Endian
- **IMG (GEM Raster)** -- Big Endian
- **JPEG** -- Big Endian
- **FLI (Autodesk Animator)** -- Little Endian
- **MacPaint** -- Big Endian
- **PCX (PC Paintbrush)** -- Little Endian
- **PostScript** -- Not Applicable (text!)
- **QTM (Quicktime Movies)** -- Little Endian (on a Mac!)
- **Microsoft RIFF (.WAV & .AVI)** -- Both
- **Microsoft RTF (Rich Text Format)** -- Little Endian
- **SGI (Silicon Graphics)** -- Big Endian
- **TIFF** -- Both, Endian identifier encoded into file
- **WPG (WordPerfect Graphics Metafile)** -- Big Endian (on a PC!)