Number Representation and Arithmetic

Stefan Mangard

October 15, 2019

Computer Organization and Networks
Graz University of Technology

Motivation and Context

How to Represent Information?

- Numbers
- Text
- Photos
- Audio
- Video
- Executables
- …

Number Representation

- During history, there have been different approaches to counting numbers

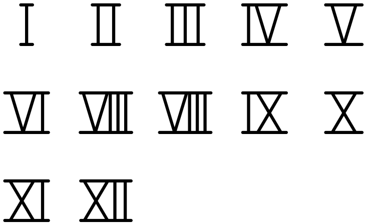
I II III IV V

VI VIII VIII IX X

XI XII

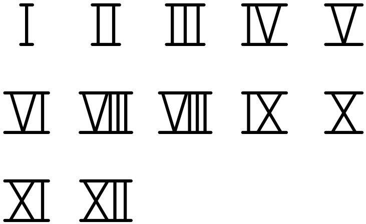
Number Representation

- During history, there have been different approaches to counting numbers
- In each approach, there are symbols to represent a certain amount of elements



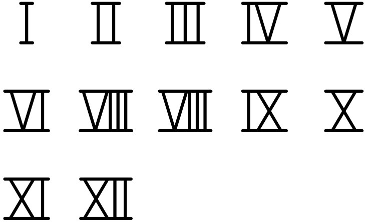
Number Representation

- During history, there have been different approaches to counting numbers
- In each approach, there are symbols to represent a certain amount of elements
- The critical question for each approach is: How to handle the situation when running out of symbols? How to define the rules to combine symbols in order to represent numbers beyond the number of symbols.



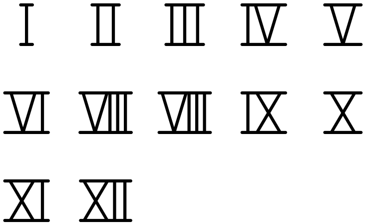
Number Representation

- During history, there have been different approaches to counting numbers
- In each approach, there are symbols to represent a certain amount of elements
- The critical question for each approach is: How to handle the situation when running out of symbols? How to define the rules to combine symbols in order to represent numbers beyond the number of symbols.
- Approach by the Romans: I, II, III, IV, V, VI, ...



Number Representation

- During history, there have been different approaches to counting numbers
- In each approach, there are symbols to represent a certain amount of elements
- The critical question for each approach is: How to handle the situation when running out of symbols? How to define the rules to combine symbols in order to represent numbers beyond the number of symbols.
- Approach by the Romans: I, II, III, IV, V, VI, ...
- Observe: There is no symbol for zero. The powers of 10 have dedicated symbols following a given patterns.



The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century

The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero

The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero
- Basic idea:

The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero
- Basic idea:
    - There are b symbols (one symbol for zero, $b-1$ symbols for non-zero numbers)

The Positional Number System

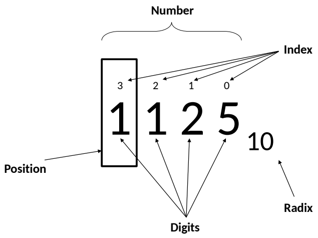
- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero
- Basic idea:
  - There are b symbols (one symbol for zero, $b - 1$ symbols for non-zero numbers)
  - Intuition: ”When running out of symbols at one position, increment the symbol left of the current symbol”

The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero
- Basic idea:
  - There are b symbols (one symbol for zero, $b - 1$ symbols for non-zero numbers)
  - Intuition: ”When running out of symbols at one position, increment the symbol left of the current symbol”
  - The sequence of n symbols $p_{n-1} \ldots p_0$ corresponds to the following value: $value = p_{n-1} \times b^{n-1} + \ldots p_1 \times b^1 + p_0 \times b^0$

The Positional Number System

- Introduced by Al-Chwarizmi in the 9th century
- The important innovation is a symbol for zero
- Basic idea:
  - There are b symbols (one symbol for zero, $b-1$ symbols for non-zero numbers)
  - Intuition: "When running out of symbols at one position, increment the symbol left of the current symbol"
  - The sequence of n symbols $p_{n-1} \ldots p_0$ corresponds to the following value:
    $value = p_{n-1} \times b^{n-1} + \ldots p_1 \times b^1 + p_0 \times b^0$



MichelBakni [CC BY-SA 4.0]

Our Number System

- Decimal system (base is “ten”) :
  - ...
  - Hundredth 10^{-2}
  - Tenth 10^{-1}
  - Ones 10^{0}
  - Tens 10^{1}
  - Hundreds 10^{2}
  - Thousands 10^{3}
  - ...
- Ten symbols: 0, 1, 2, ..., 9
- With n positions: 10^n possible different values
- $2345 = 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0$
- $345.67 = 3 \cdot 100 + 4 \cdot 10 + 5 \cdot 1 + 6 \cdot \frac{1}{10} + 7 \cdot \frac{1}{100}$

Base b

- Number system with base b:
  - ... b^3 b^2 b^1 b^0 . b^{-1} b^{-2} b^{-3}
  - b Symbols

Base b = 2

- Two symbols: 0, 1
- With n positions: 2^n possible different values
- $(1011)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$
- $(011.01101)_2 = 2 + 1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{32}$
    - ...
    - Quarters 2^{-2}
    - Halves 2^{-1}
    - Ones 2^0
    - Twos 2^1
    - Fours 2^2
    - Eights 2^3
    - ...

Counting with Binary Numbers

- $(0000)_2 = 0 + 0 + 0 + 0 = 0$
- $(0001)_2 = 0 + 0 + 0 + 1 = 1$
- $(0010)_2 = 0 + 0 + 2 + 0 = 2$
- $(0011)_2 = 0 + 0 + 2 + 1 = 3$
- $(0100)_2 = 0 + 4 + 0 + 0 = 4$
- ..
- $(1101)_2 = 8 + 4 + 0 + 1 = 13$
- $(1110)_2 = 8 + 4 + 2 + 0 = 14$
- $(1111)_2 = 8 + 4 + 2 + 1 = 15$

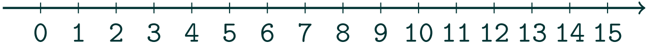
Translation between Binary, Hexadecimal and Decimal

1011 1100 0011 0101

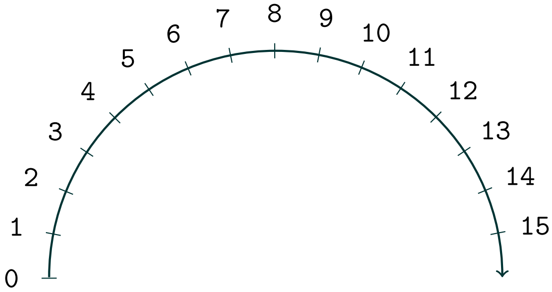
B C 3 5

Hex.	Dualsystem				Dez.
0	0	0	0	0	00
1	0	0	0	1	01
2	0	0	1	0	02
3	0	0	1	1	03
4	0	1	0	0	04
5	0	1	0	1	05
6	0	1	1	0	06
7	0	1	1	1	07
8	1	0	0	0	08
9	1	0	0	1	09
A	1	0	1	0	10
B	1	0	1	1	11
C	1	1	0	0	12
D	1	1	0	1	13
E	1	1	1	0	14
F	1	1	1	1	15

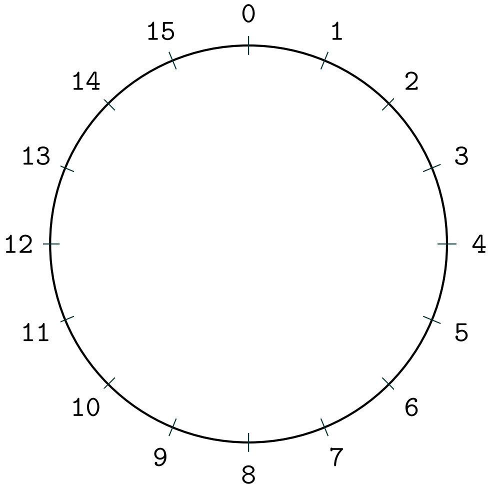
Number line



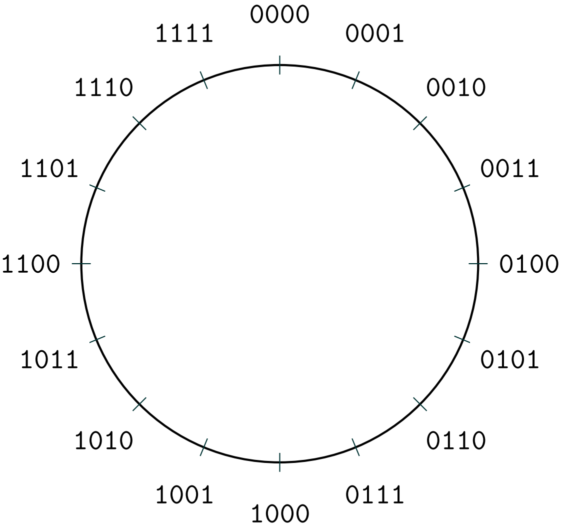
Number line



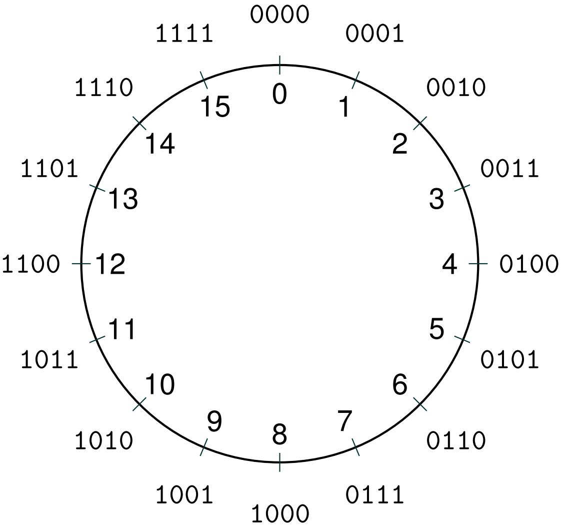
Number line



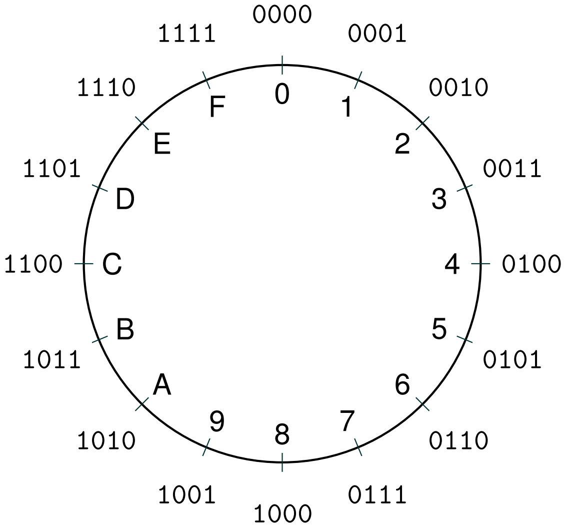
Binary “numbers” with 4 bits (sorted)



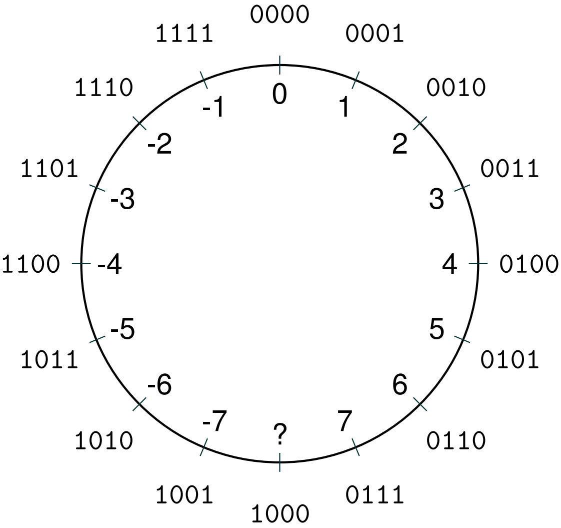
Interpretation as Unsigned Numbers



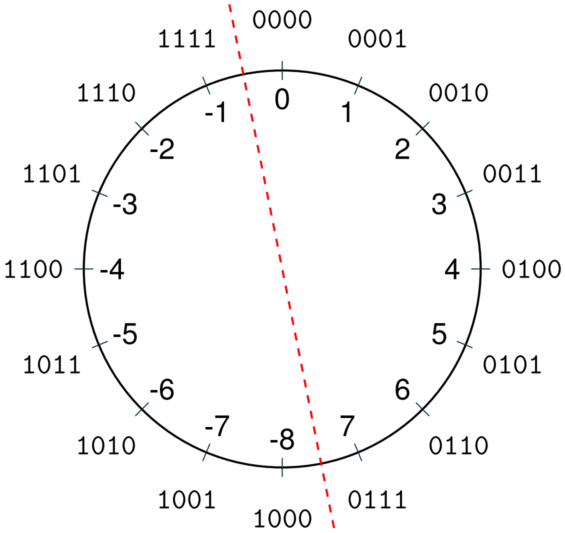
Abbreviation With Hexadecimal Numbers



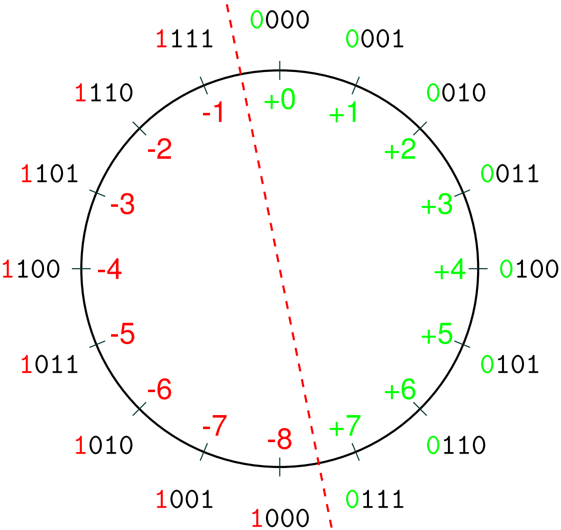
We Can Also Count in the Other Direction



Two Halves: Positive and Negative



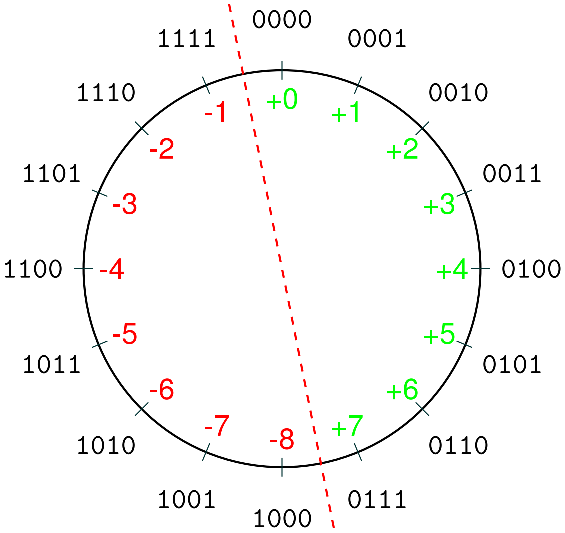
Interesting "side effect": The most significant bit (MSB) denotes the sign



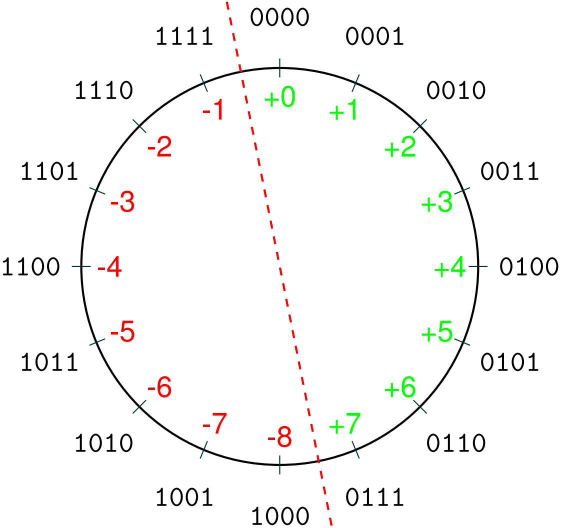
Interesting "side effect": The most significant bit (MSB) denotes the sign

- Important: The most significant bit is not the sign in the traditional sense. Here, we do not have the usual representation as sign followed by magnitude of number. However, the most significant bit indicates whether the number is positive or negative.

“Signed” Representation

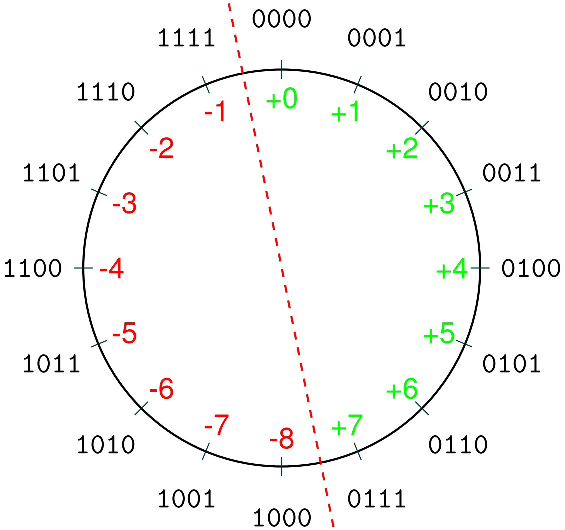


"Two's Complement Representation"



How to subtract?

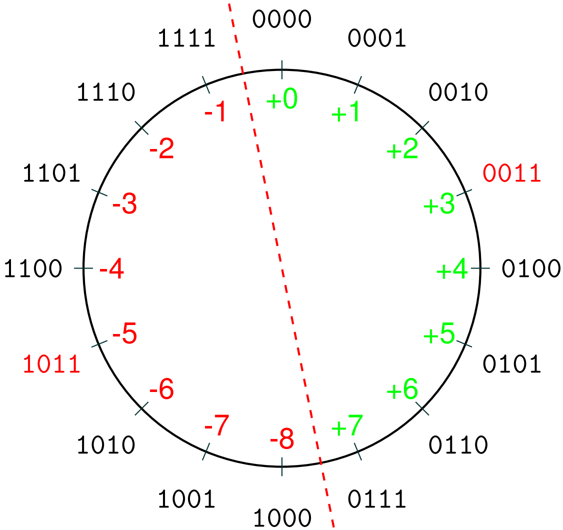
3 - 5



3 - 5

- (+3) + (-5)

Subtraction is also an addition:
We add the negation of 5

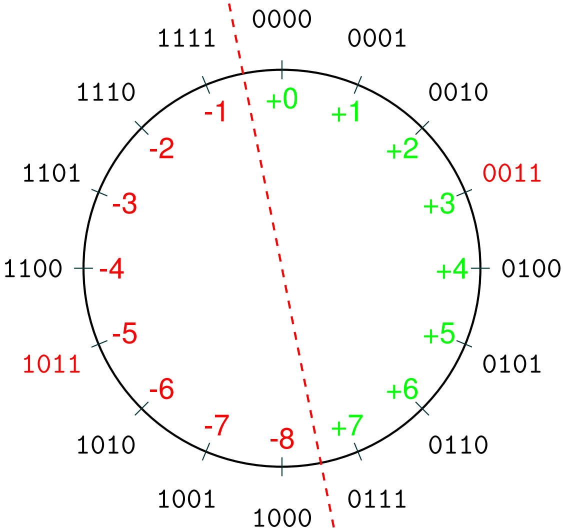


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1

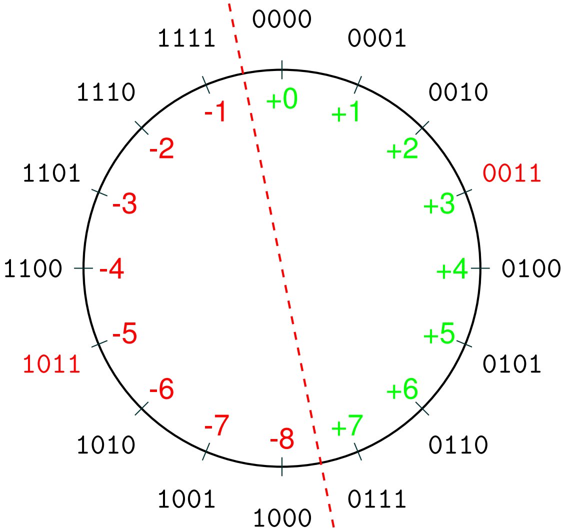


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1
			1	0

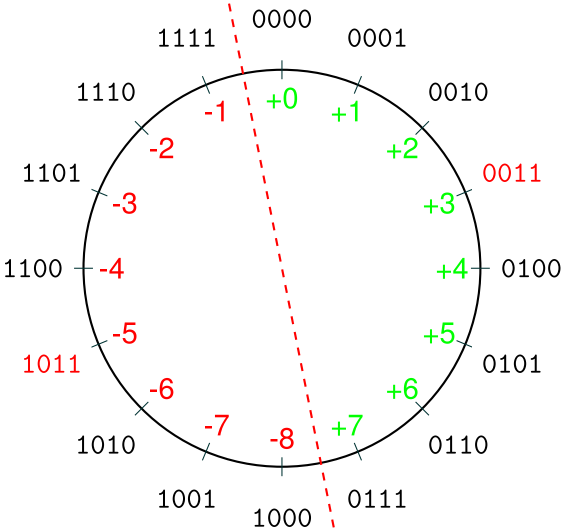


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1
		^{1}	1^{1}	0

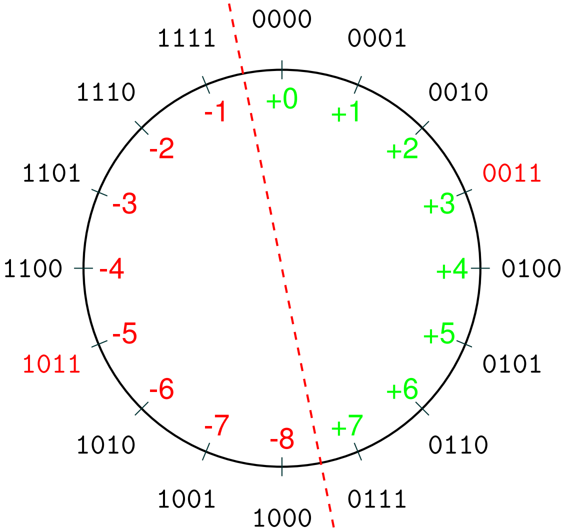


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1
		1^1	1^1	0

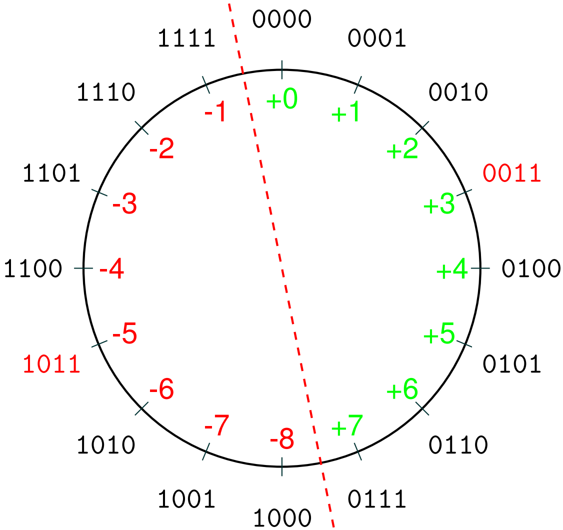


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1
	1	1	1	0

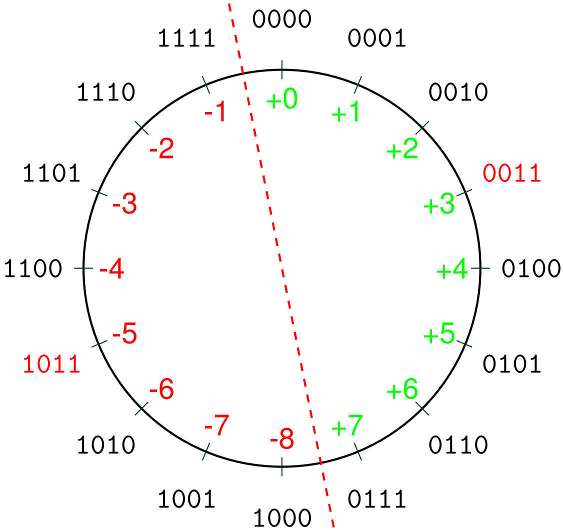


How to subtract?

3 - 5

- (+3) + (-5)

	0	0	1	1
+	1	0	1	1
	1	1	1	0

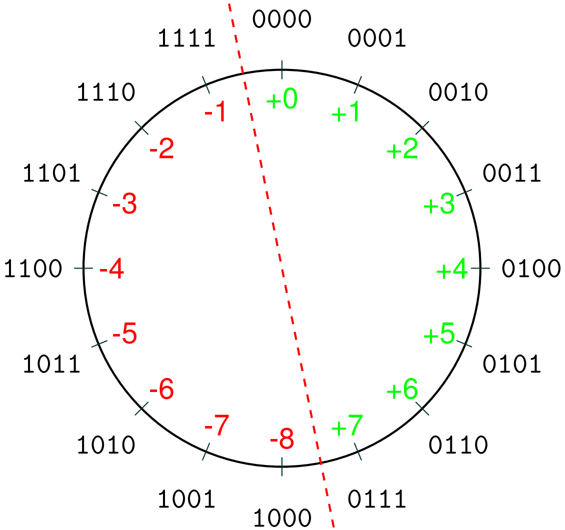


What Happens If We Leave the Interval: Overflow

3 + 6

- (+3) + (+6) = (-7)?

	0	0	1	1
+	0	1	1	0
	1	0	0	1



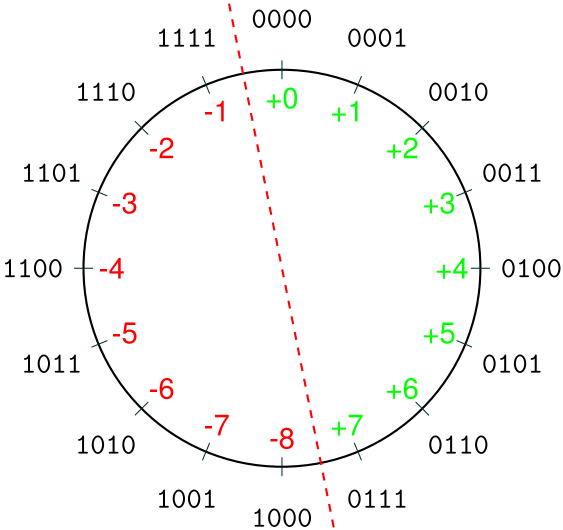
What Happens If We Leave the Interval: Underflow

3 + 6

- (-3) + (-6) = (+7)?

	1	1	0	1
+	1	0	1	0
~~1~~	0	1	1	1

There is no 5 bit in this 4 bit world



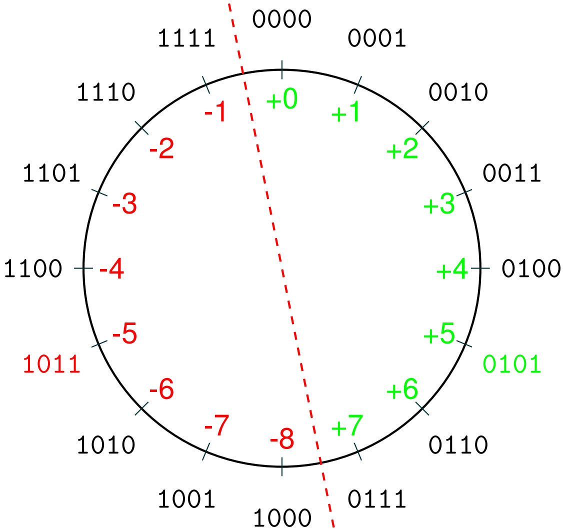
Negation of a Value: We do the "two's complement"

+5 : 0 1 0 1 → Invert all bits: "One's complement"

	1	0	1	0
+	0	0	0	1
	1	0	1	1

adding 1 makes the "two's complement"

-5 : 1 0 1 1



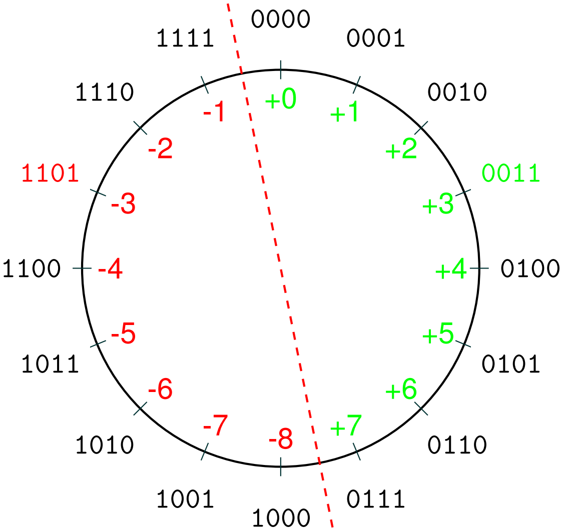
Two's Complement Works Both Ways

-3 : 1 1 0 1 → Invert all bits: "One's complement"

	0	0	1	0
+	0	0	0	1
	0	0	1	1

adding 1 makes the "two's complement"

+3 : 0 0 1 1



Let’s Check the Number Representation on CPUs (Intel, ARM, RISC-V)

- Compile and run example 3 of chapter 2 in order to explore the overflow and underflow behavior of different data types on your CPU

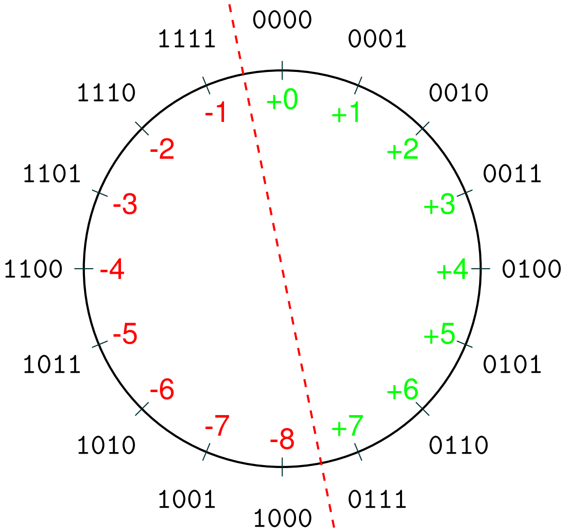
Rational Numbers

- To the right of the decimal point:
  - Thenth 10^{-1}
  - Hundredth 10^{-2}
  - Thousandth 10^{-3}
  - ...
- Example: $(0.1234)_{10}$
- Multiply with 10^4 leads to $(1234)_{10}$

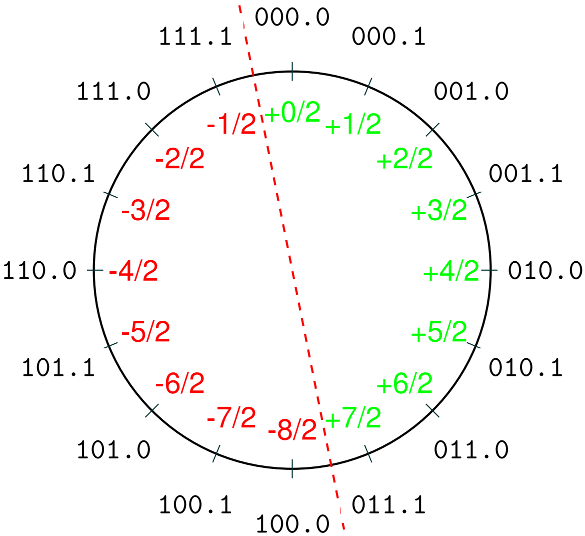
Rational Numbers With Base 2

- To the right of the binary point:
  - Halves 2^{-1}
  - Quarters 2^{-2}
  - Eighths 2^{-3}
  - ...
- Example: $(0.1101)_2 = \frac{1}{2} + \frac{1}{4} + \frac{1}{16} = \frac{13}{16}$
- Alternative view: Multiply $(0.1101)_2$ with 2^4 and you get $(1101)_2$ i.e $(13)_{10}$. The multiplication with 2^4 shifts the "binary point" by 4 positions to the left. Thus, $(0.1101)_2$ is the same as $\frac{13}{16}$.

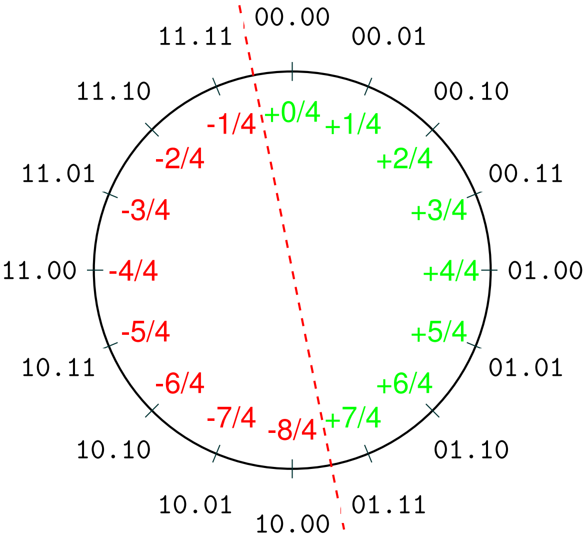
Rational Numbers in “signed” Format



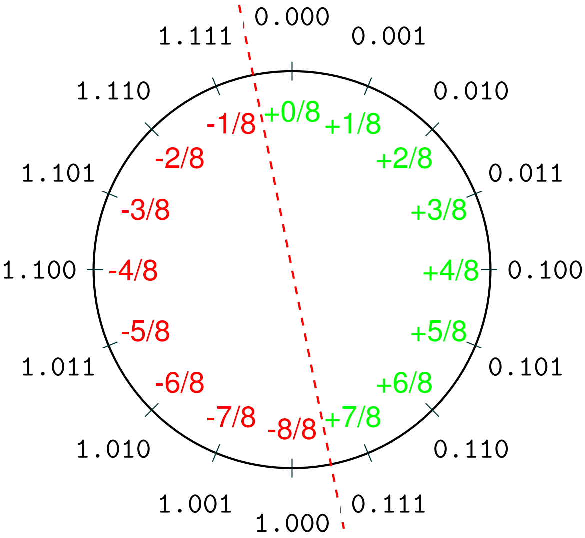
Division by 2



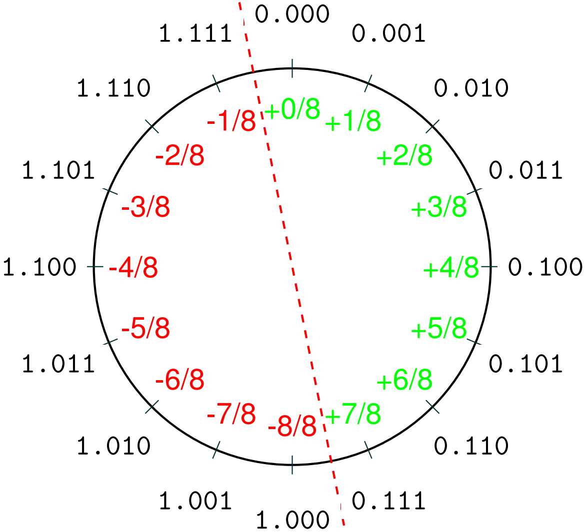
Division by 4



Division by 8



We cannot divide by 16 since we need at least 1 bit left of the “point”



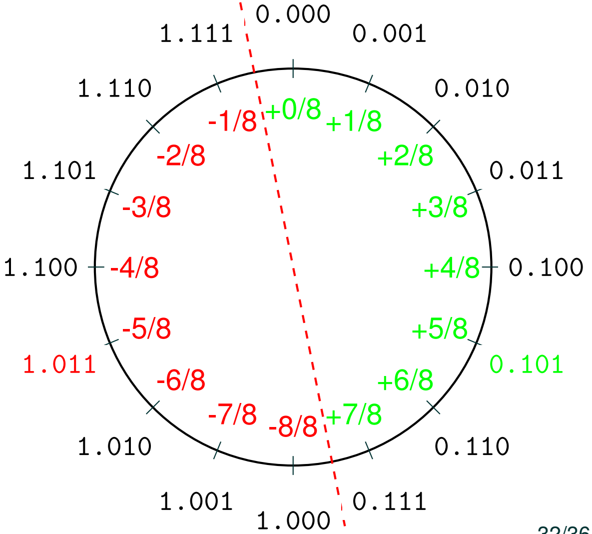
Two's Complement Again

-5/8 : 1 . 0 1 1 → Invert all bits: "One's complement"

	0.	1	0	0
+	0.	0	0	1
	0.	1	0	1

adding 1 makes the "two's complement"

+5/8 : 0. 1 0 1



Two's Complement Again

- Alternative view: Multiply first $-\frac{5}{8}$ with 2^3. This corresponds to a shift of the binary point by three positions to the right. The result is -5. Next, we compute the twos complement of -5 and get +5. Finally, we divide this value by 2^3 again. This corresponds to a shift of the point by 3 positions to the left. As a result we get $+\frac{5}{8}$.

Division by the Base

- Right shift by one position: 123.4 / 10 = 12.34
- Arithmetic Shift Right (ASR): All bits get shifted to the right by one position. The value of the most significant bit gets duplicated.
- Logic Shift Right (LSR): All bits get shifted to the right by one position. The most significant bit gets 0.
- Division by 2:
  - Unsigned: LSR: Half of $(0110)_2$ is $(0011)_2$
  - Signed: ASR: Half of $(-6)_{10} = (1010)_2$ is $(1101)_2$
  - Half of $(-\frac{4}{8}) = (1.100)_2 = (1.110)_2 = -\frac{2}{8}$

Division by the Base

- Watch out: negative numbers are rounded towards “lower values”
  - It does not matter by how many positions you shift to the right: $(1111)_2$ stays always $(1111)_2$.
  - This problem can be found throughout the history of computing. Check out `https://en.wikipedia.org/wiki/Arithmetic_shift`

Verilog and Number Representation

- Verilog assumes Two's Complement representation whenever arithmetic operations are done in the Verilog code
- All arithmetic operations are available
- Compile and run example 1 and example 2 of chapter 2 in order to see how to build an Arithmetic Logic Unit (ALU)

Number Representation and Arithmetic

Stefan Mangard

October 15, 2019

Computer Organization and Networks
Graz University of Technology