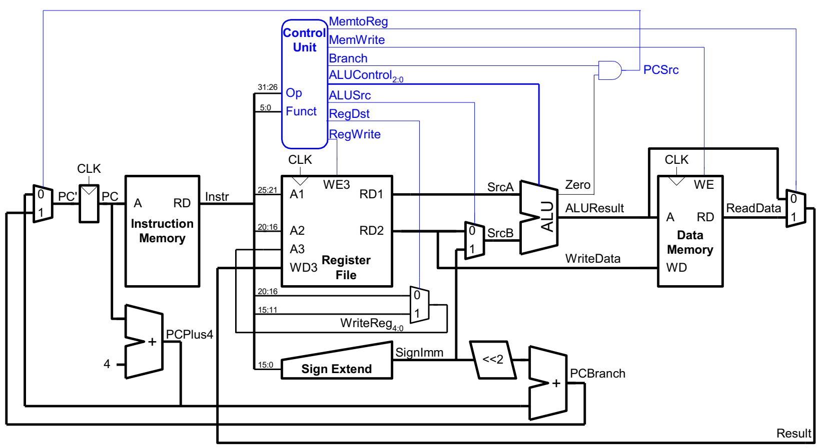
Number representations

Becker/Molitor, Chapter 3.3

Jan Reineke
Universität des Saarlandes

Roadmap: Computer architecture



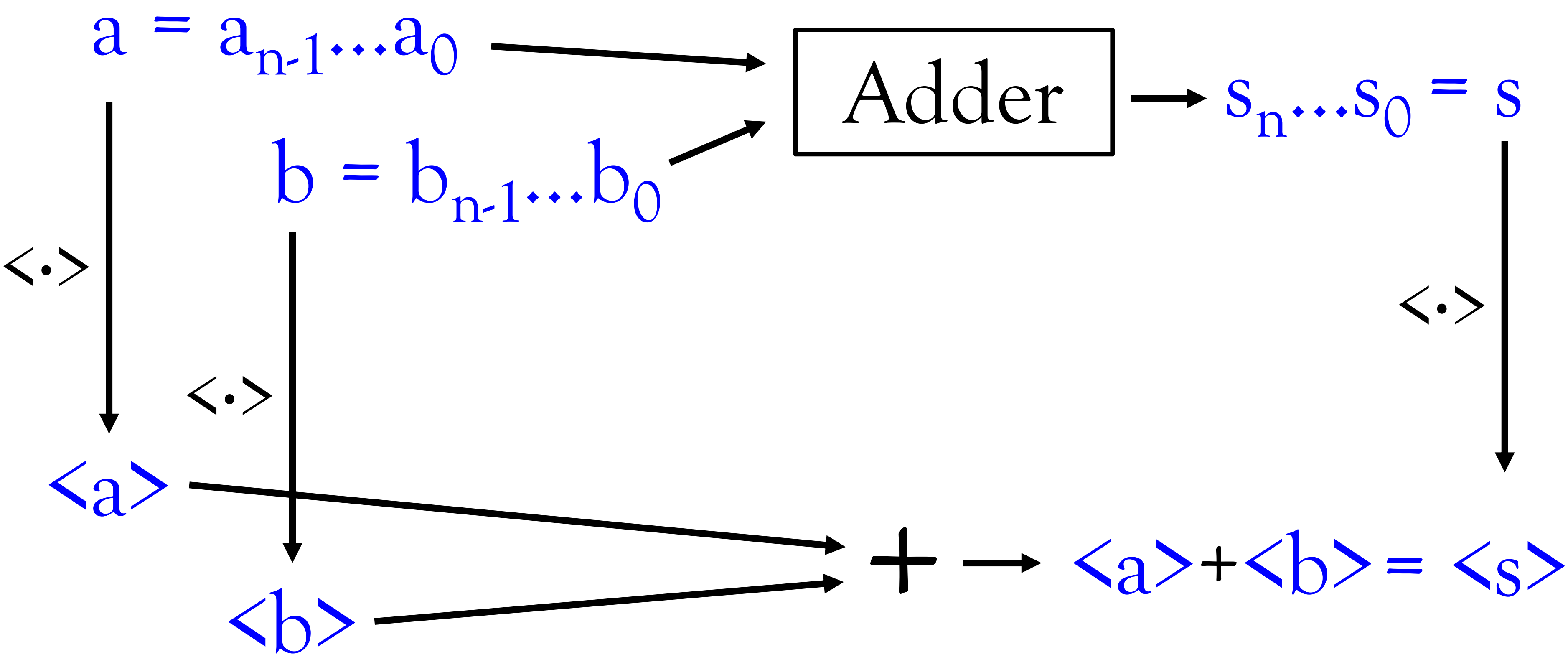
1. Combinatorial circuits: Boolean Algebra/Functions/Expressions/Synthesis
2. **Number representations**
3. Arithmetic Circuits: Addition, Multiplication, Division, ALU

4. Sequential circuits: Flip-Flops, Registers, SRAM, Moore and Mealy automata
5. Verilog

6. Instruction Set Architecture
7. Data path & Control path

8. Performance: RISC vs. CISC, Pipelining, Memory Hierarchy

Outlook: Arithmetic circuits



Challenge: Number representations

Internally, computers represent numbers by binary strings of some fixed length n bits.

Questions:

1. How many different numbers can be represented?
2. How to represent *natural numbers*?
3. How to represent *integers*?
   Challenge: negative numbers
4. How to represent *rational numbers*?

fixed-point numbers

5. How to represent *very large* and very *small numbers*?

floating-point numbers

1. How many different numbers can be represented?

For n bits and b (*typically* $b=2$) different numerals in each position,

- there are b^n distinct strings, and so
- **at most** b^n distinct numbers can be represented, e.g. $0, \ldots, b^n-1$ or $-b^{n-1}, \ldots, b^{n-1}-1$

Examples of numeral systems

Examples:

- **Binary numeral system**

  b=2, $Z = \{0,1\}$

- **Decimal numeral system**

  b=10, $Z = \{0,1,2,3,4,5,6,7,8,9\}$

- **Hexadecimal numeral system:**

  b=16 $Z = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

Numeral systems **formally**

Definition (Positional numeral system):

A **positional numeral system** is a triple $S = (b, Z, \delta)$ with the following properties:

- $b \in \mathbb{N}$ is a natural number, the **basis**.
- Z is a set of symbols of size b, the **numerals** (**or digits**).
- $\delta : Z \rightarrow \{0, 1, \ldots, b\text{-}1\}$ is a bijective mapping that associates each numeral with a natural number between 0 and $b\text{-}1$.

2. Representation of **natural numbers**

Which natural number <d> is represented the sequence

$d = d_n d_{n-1} \ldots d_1 d_0$

of numerals from a positional numeral system (b, Z, δ)?

Examples:

Let d = 0110

- $b = 2, \quad <d> =$
- $b = 10, \quad <d> =$
- $b = 16, \quad <d> =$

In general: $< d > = < d_n d_{n-1} \ldots d_1 d_0 > = \sum_{i=0}^{n} b^i \cdot \delta(d_i)$

Binary numbers

For $b = 2$ and $n = 2$ we thus have:

d	000	001	010	011	100	101	110	111
<d>	0	1	2	3	4	5	6	7

Properties:

- Smallest representable number: 0
- Largester representable number: $2^{n+1}-1$
- „Adjacent numbers“ are at distance 1.

3. Representing **integers**, in particular negative numbers

Goals:

1. Want to cover large number space:
   → aim for *unique* number representation
2. Would like to reuse arithmetic circuits for *natural numbers*

Signed magnitude representation

1. Approach: **Signed magnitude** representation.

The most significant digit d_n determines the **sign** of the number:

$$[d_n d_{n-1} \ldots d_0]_{SM} := (-1)^{d_n} \cdot \langle d_{n-1} \ldots d_0 \rangle$$
$$= (-1)^{d_n} \cdot \Sigma_{i=0,\ldots,n-1}\, d_i \cdot 2^i.$$

d	000	001	010	011	100	101	110	111
$[d]_{BV}$	0	1	2	3	0	-1	-2	-3

Signed magnitude representation

$$[d_n d_{n-1} \ldots d_0]_{SM} := (-1)^{d_n} \cdot \Sigma_{i=0,\ldots,n-1}\, d_i \cdot 2^i$$

d	000	001	010	011	100	101	110	111
$[d]_{SM}$	0	1	2	3	0	-1	-2	-3

Properties:

- The number range is **symmetric**:
  - Smallest number: $-(2^n-1)$
  - Largest number: 2^n-1
- To invert a number d, one needs to **flip the first bit**.
- Two representations of zero (000 and 100 for n=2).
- „Adjacent numbers“ are at distance 1 in terms of absolute value.

(2^n-1) complement = One's complement

2. Approach: Representation via **(2^n-1) complement**.

The most-significant digit d_n *again* determines whether it is a positive or a negative number.

But now $d_n \cdot (2^n-1)$ is subtracted:

$$[d_n d_{n-1} \ldots d_0]_1 := \langle d_{n-1} \ldots d_0 \rangle - d_n \cdot (2^n-1)$$
$$= \sum_{i=0,\ldots,n-1} d_i \cdot 2^i - d_n \cdot (2^n-1).$$

d	000	001	010	011	100	101	110	111
$[d]_1$	0	1	2	3	-3	-2	-1	0

One's complement

$$[d_n d_{n-1} \ldots d_0]_1 := \Sigma_{i=0,\ldots,n-1}\, d_i \cdot 2^i - d_n \cdot (2^n - 1)$$

d	000	001	010	011	100	101	110	111
$[d]_1$	0	1	2	3	-3	-2	-1	0

Properties:

- The number range is **symmetric**:
  - Smallest number: $-(2^n-1)$
  - Largest number: 2^n-1
- To invert a number d, one needs to **flip all bits.**
- Two representations of zero (000 and 111 for n=2).
- „Adjacent numbers" are at distance 1 ~~in terms of absolute value~~.

2^n complement = Two's complement

3. *Approach:* Representation via **2^n complement**.

The most-significant digit d_n *again* determines whether it is a positive or a negative number.

But now $d_n \cdot 2^n$ is subtracted:

$$[d_n d_{n-1} \ldots d_0]_2 := \langle d_{n-1} \ldots d_0 \rangle - d_n \cdot 2^n$$
$$= \Sigma_{i=0,\ldots,n-1}\, d_i \cdot 2^i - d_n \cdot 2^n.$$

d	000	001	010	011	100	101	110	111
$[d]_2$	0	1	2	3	-4	-3	-2	-1

Two's complement

$$[d_n d_{n-1} \ldots d_0]_2 := \Sigma_{i=0,\ldots,n-1}\, d_i \cdot 2^i - d_n \cdot 2^n$$

d	000	001	010	011	100	101	110	111
$[d]_2$	0	1	2	3	-4	-3	-2	-1

Properties:

- The number range is **asymmetric**:
  - Smallest number: -2^n
  - Largest number: 2^n-1
- Let d be arbitrary and d' be obtained by flipping all digits of d. Then we have $[d']_2+1 = -[d]_2$.
- The number representation is **unique**.
- „Adjacent numbers" are at distance 1 ~~in terms of absolute value~~.

Two’s complement

Main advantage of two’s complement:
Can use arithmetic circuits for additions of natural numbers also for integers.

(→ more details later)

4. Representing **rational numbers**

1. *Approach:* **Fixed-point numbers**.

- Interpret first part of the digit sequence as integral part and the rest as decimal places.
- Assume we have n+1 integral and k decimal places.
- Then the value <d> of a non-negative fixed-pointer number

$$d = d_n d_{n-1} \ldots d_1 d_0 \text{ , } d_{-1}, \ldots, d_{-k}$$

is given by

$$< d > = < d_n d_{n-1} \ldots d_1 d_0 , d_{-1}, \ldots, d_{-k} > = \sum_{i=-k}^{n} b^i \cdot \delta(d_i)$$

Negative fixed-point numbers: Two's complement

Extension of two's complement to fixed-point numbers:

$$\boxed{[d_n d_{n-1} \ldots d_0, d_{-1} \ldots d_{-k}]_2 := \Sigma_{i=-k,\ldots,n-1}\, d_i \cdot 2^i - d_n \cdot 2^n}$$

Problems with fixed-point numbers

Consider the set of numbers that have a two's complement representatoin with *n* integral and *k* decimal places.

- **Cannot represent very large nor very small numbers**!
  - Largest numbers in terms of absolute value: -2^n and 2^n-2^{-k}
  - Smallest non-zero numbers in terms of absolute value: -2^{-k} and 2^{-k}
- Representation is **not** closed under **addition/substraction**!
  - $2^{n-1}+2^{n-1}$ is not representable even though the operands are representable → Overflow

4. Representing **rational numbers**

2. *Approach:* **Floating-point numbers.**

Position of the decimal point is not fixed, it is “floating”.

Covering a larger number range using the same number of digits.

Single precision floating-point numbers: $(-1)^S \cdot M \cdot 2^E$

31	30 29 28 27 26 25 24 23	22 21 20 19 ... 3 2 1 0
S	Exponent E	Mantissa M

float

Double precision floating-point numbers: $(-1)^S \cdot M \cdot 2^E$

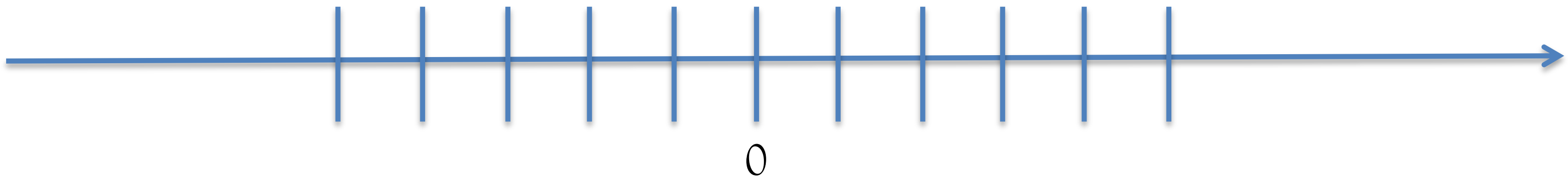
63	62 61 60 ... 54 53 52	51 50 49 ... 3 2 1 0
S	Exponent E	Mantissa M

double

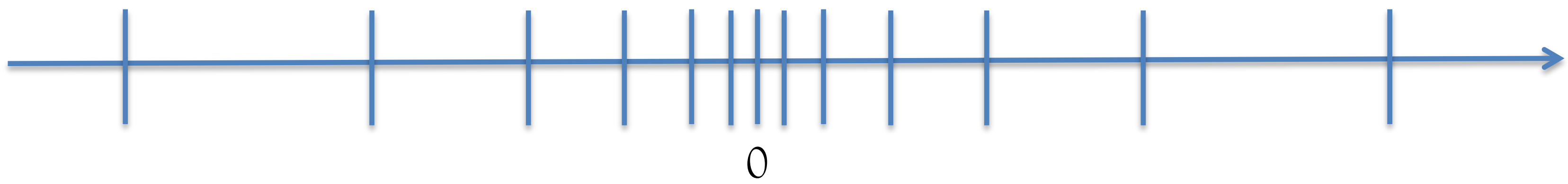
It remains to define how the mantissa and exponent bits are interpreted.
This is e.g. captured by the IEEE 754 standard.

Advantages of floating-point numbers

Distribution of fixed-point numbers:



Distribution of floating-point numbers:



- In the fixed-point representation the distance between adjacent numbers is **the same everywhere**.
- In the floating-point representation the relative difference between adjacent numbers is kept **small**.

Problems with floating-point numbers

- Associativity does not hold:

$$\varepsilon = \varepsilon + \underbrace{(1+(-1))}_{0} \neq \overbrace{\underbrace{(\varepsilon+1)}}^{1} + (-1) = 0$$

small number

$1 = 1.000\ldots0$

$\varepsilon = 0.0\ldots0\ldots01$

- Distributivity holds neither