Data representation

Binary numbers can be represented in two forms:

- fixed point representation
- floating point representation

We also need to represent negative and positive numbers for computations. The signed binary numbers can be represented in 3 systems:

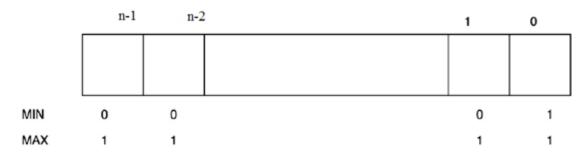
- signed magnitude system
- one's complement system
- two's complement system

Binary Numbers Representation

Fixed-point reprezentation

Unsigned numbers

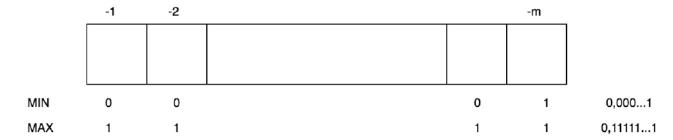
1. Integer numbers



$$1 \le N \le 2^n - 1$$

Decimal point is considered to be after the rightmost digit, but is not stored in the register.

2. Fractional numbers



$$2^{-m} \le N \le 1 - 2^{-m}$$

Signed numbers

The signed binary numbers can be represented in 3 systems:

- signed magnitude system (a direct code)
- one's complement system (an indirect code)
- two's complement system (a complement code)
 In all these systems an extra bit position is used to represent the sign.

The MSB of a bit string is used as the sign (0=plus, 1=minus).

Signed – magnitude system (Direct Code)

Integers
$$N=$$

$$\begin{cases} 0 \ b_{n-1} \ b_{n-2} \ b_{n-3} \dots \ b_1 \ b_0; \ N \ge 0 \\ 1 \ b_{n-1} \ b_{n-2} \ b_{n-3} \dots \ b_1 \ b_0; \ N \le 0 \end{cases}$$

	n	n-1	n-2	n-3	 1	0	
	sg	b_{n-1}	b_{n-2}	b_{n-3}	 b_1	b_0 .	binary point
N	MSB					LSB	

Example: for 8 bits

$$N=-17_{10}$$
 $N_{DC}=10010001$
 $N=+40_{10}$ $N_{DC}=00101000$

$$-(2^{n}-1) \le N_{DC} \le 2^{n}-1$$

Fractionals

$$-(1-2^{-m}) \le N_{DC} \le (1-2^{-m})$$

Two's complement system (Complement Code) r^n -N.

Integers

$$\mathbf{N}_{\text{CC}} = \begin{cases} 0 \ b_{n-1} b_{n-2} b_{n-3} ... b_1 b_0 \ ; \ \mathbf{N} \ge 0 \\ 1 \ \overline{b_{n-1}} \overline{b_{n-2}} \overline{b_{n-3}} ... \overline{b_1} (\overline{b_0} + 1); \ \mathbf{N} \le 0 \end{cases}$$

For positive numbers the complemented code representation corresponds to the direct code. For negative numbers the CC is obtained by complementing and adding 1 to the result. Example:

$$-2^{n} \le N_{CC} \le 2^{n}-1$$

For a fractional number:

$$\text{--}1 \leq N_{CC} \leq 1\text{--}2\text{--m}$$

Two's complement system (Complement Code)

Examples.

76

-123

127

-127

-128

+0

-0

One's complement system (Indirect Code) (rⁿ-1)-N.

$$N_{IC} = \begin{cases} 0 & b_{n-1}b_{n-2}b_{n-3}...b_1b_0 \\ 1 & \overline{b_{n-1}}\overline{b_{n-2}}\overline{b_{n-3}}...\overline{b_1}\overline{b_0} \end{cases}; N \le 0$$

Example:

There are two representations of 0 – the positive one (0000) and the negative one (1111). For integers the range of numbers we can represent is:

$$-(2^{n}-1) \le N_{IC} \le 2^{n}-1$$

The largest value	0	1	1	1	1
The lowest value	1	0	0	0	0

For a fractional number:

$$-(1-2^m) \le N_{IC} \le 1-2^m$$

Exercises

- Give the value of 88, 0, 1, 127, and 255 in 8-bit unsigned representation.
- Give the value of +88, -88, -1, 0, +1, -128, and +127 in 8-bit 2's complement signed representation.
- Determine the decimal value of numbers 10010110 unsigned 10010110 2's complement

Floating-Point Representation

IEEE numbers are stored using a kind of scientific notation.

± mantissa * 2 exponent

We can represent floating-point numbers with three binary fields: a sign bit s, an exponent field e, and a fraction field f.

s	е	f
---	---	---

The IEEE 754 standard defines several different precisions.

- Single precision numbers include an 8-bit exponent field and a 23-bit fraction, for a total of 32 bits.
- Double precision numbers have an 11-bit exponent field and a 52-bit fraction, for a total of 64 bits.

Mantissa

		£
S	e	I

□ There are many ways to write a number in scientific notation, but there is always a unique normalized representation, with exactly one non-zero digit to the left of the point.

$$0.232 \times 10^3 = 23.2 \times 10^1 = 2.32 \times 10^2 = \dots$$

$$01001 = 1.001 \times 2^3 = \dots$$

What's the normalized representation of 00101101.101 ? 00101101.101 = 1.01101101 × 2⁵

■ What's the normalized representation of 0.0001101001110 ?

0.0001101001110= $1.1101001111 \times 2^{-4}$

Mantissa

S	е	f

■ There are many ways to write a number in scientific notation, but there is always a unique normalized representation, with exactly one non-zero digit to the left of the point.

$$0.232 \times 10^3 = 23.2 \times 10^1 = 2.32 \times 10^2 = \dots$$

$$01001 = 1.001 \times 2^3 = \dots$$

- The field f contains a binary fraction.
- The actual mantissa of the floating-point value is (1 + f).
 - In other words, there is an implicit 1 to the left of the binary point.
 - For example, if f is 01101..., the mantissa would be 1.01101...
- A side effect is that we get a little more precision: there are 24 bits in the mantissa, but we only need to store 23 of them.
- But, what about value 0?

Exponent

s e f

- There are special cases that require encodings
 - Infinities (overflow)
 - NAN (divide by zero)
- For example:
 - Single-precision: 8 bits in e → 256 codes; 11111111 reserved for special cases → 255 codes; one code (00000000) for zero → 254 codes; need both positive and negative exponents → half positives (127), and half negatives (127)
 - Double-precision: 11 bits in e → 2048 codes; 111...1 reserved for special cases → 2047 codes; one code for zero → 2046 codes; need both positive and negative exponents → half positives (1023), and half negatives (1023)

Exponent

		_
S	е	l f

- ☐ The e field represents the exponent as a biased number.
 - It contains the actual exponent plus 127 for single precision, or the actual exponent plus 1023 in double precision.
 - This converts all single-precision exponents from -126 to +127 into unsigned numbers from 1 to 254, and all double-precision exponents from -1022 to +1023 into unsigned numbers from 1 to 2046.
- Two examples with single-precision numbers are shown below.
 - If the exponent is 4, the e field will be $4 + 127 = 131 (10000011_2)$.
 - If e contains 01011101 (93₁₀), the actual exponent is 93 127 = 34.
- Storing a biased exponent means we can compare IEEE values as if they were signed integers.

Mapping Between e and Actual Exponent

е		Actual Exponent
0000 0000		Reserved
0000 0001	1-127 = -126	-126 ₁₀
0000 0010	2-127 = -125	-125 ₁₀
0111 1111		0 ₁₀
1111 1110	254-127=127	127 ₁₀
1111 1111		Reserved

Special Values (single-precision)

E	F	meaning	Notes
0000000	00	0	+0.0 and -0.0
0000000	XX	Valid number	Unnormalized =(-1) ^S x 2 ⁻¹²⁶ x (0.F)
11111111	00	Infinity	
11111111	XX	Not a Number	

Converting an IEEE 754 number to decimal

s	е	f

The decimal value of an IEEE number is given by the formula:

- Here, the s, f and e fields are assumed to be in decimal.
 - (1 2s) is 1 or -1, depending on whether the sign bit is 0 or 1.
 - We add an implicit 1 to the fraction field f, as mentioned earlier.
 - Again, the bias is either 127 or 1023, for single or double precision.

Example IEEE-decimal conversion

- Let's find the decimal value of the following IEEE number.
- First convert each individual field to decimal.
 - The sign bit s is 1.
 - The e field contains $011111100 = 124_{10}$.
 - The mantissa is $0.11000... = 0.75_{10}$.
- Then just plug these decimal values of s, e and f into our formula.

This gives us $(1-2) * (1+0.75) * 2^{124-127} = (-1.75 * 2^{-3}) = -0.21875$.

Converting a decimal number to IEEE 754

- What is the single-precision representation of 347.625?
 - 1. First convert the number to binary: $347.625 = 101011011.101_2$.
 - Normalize the number by shifting the binary point until there is a single 1 to the left:

```
101011011.101 \times 2^{0} = 1.01011011101 \times 2^{8}
```

- The bits to the right of the binary point comprise the fractional field f.
- The number of times you shifted gives the exponent. The field e should contain: exponent + 127.
- 5. Sign bit: 0 if positive, 1 if negative.

Exercise

■ What is the single-precision representation of 639.6875

$$639.6875 = 10011111111.10112$$
$$= 1.0011111111111111 \times 29$$

The single-precision representation is: 0 10001000 001111111110110000000000