

Computer Science Basics

Negative numbers

Fall Term 2025/2026

Emmanuel Benoist | BFH-TI

Negative Numbers

- ▶ Last Week
- ▶ Negative Numbers
- ▶ Possible representations
- ▶ Two's Complement
- ▶ Conclusions

Last Week

Counting with bases 2 and 16

- **Computers work only in binary**
 - ▣ Notations in Binary are too long
 - ▣ We use hex to represent binary values
- **You should be familiar with hex notation**
 - ▣ It is the center of low level programming.
 - ▣ One solution: do the exercises!

Negative Numbers

How to Represent Negative Numbers?

- **Numbers can be positive**
 - Decimal: 15
 - Stored in memory in hexadecimal: 0x0F (on 8 bits)
 - or 0x0000000F in 32 bit
- **And can also be negative**
 - Decimal: -15
 - How should we represent it?

Possible representations

Different possible representations

- **One bit for the sign**

- The first bit is used for the sign
- 15 is represented by 0x000F on a 16-bit architecture
- -15 by 0x800F

- **Problems:**

- Two representations of 0 exist: 0x0000 and 0x8000
- Standard addition does not work:
 $15 + (-15) = 0x000F + 0x800F = 0x801E = -30$

- **One's Complement**

- The negative number is obtained by inverting all the bits
- 15 = 0x000F
- 15 = 0b 0000 0000 0000 1111
- -15 is represented by 0b 1111 1111 1111 0000
- -15 is represented by 0xFFFF

- **Problems**

- We have two different representations of 0 0x0000 and 0xFFFF
- Addition does not work

$15 = 0x000F$ and $-5 = 0xFFFFA$ The sum is $0x10009$ which makes 9 if we forget the

Two's Complement

Two's Complement

- **Negative numbers are designed to respect addition**

- $15 + (-10) = 5$
- Idea: $-X$ is represented by $2^n - X$ if n is the number of bits
- $X + (-X) = X + 2^n - X = 0 + \text{overload}$

- **Example 1**

- Notation for -1 on 16 bits
- $2^{16} - 1 = 2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{11} + \dots + 2^1 + 2^0$
- -1 is written `0xFFFF`
- $1 + (-1) = \text{0x10000}$ (overload is out of the 16 bits)

- **Example 2**

- Notation for -20 on 16 bits
- $2^{16} - 20 = \text{Hard to compute}$

Two's Complement

■ Method for computing

- Computing the representation of $-X$
- Take the binary representation of X
- Invert all the bits of X
- Add one

■ Example 2 (Cont): -20

- $20 = \text{ob } 0000\ 0000\ 0001\ 0100$
- We invert all the bits
- $\text{ob } 1111\ 1111\ 1110\ 1011$
- We add one
- $\text{ob } 1111\ 1111\ 1110\ 1100$
- Which can be noted in hexa: oxFFEC

■ Example 3 : -32

- $32 = \text{ob } 0000\ 0000\ 0010\ 0000$
- it makes $\text{ob } 1111\ 1111\ 1101\ 1111$
- We add one: $\text{ob } 1111\ 1111\ 1110\ 0000$
- Representation = ox FFE0

Conclusions

Conclusion

- **Numbers are stored in binary in computers**
 - Easy for positive numbers
- **Two's Complement Notation**
 - Works for positive and negative numbers
 - Uses the same addition for signed and unsigned numbers
- **How to represent real numbers?**
 - Topic of next course.