Base 2

Signed numbers

Assembly language programming
By xorpd

xorpd.net

Objectives

- You will recall the traditional ways to represent negative numbers.
- You will learn how to invoke “subtraction by addition” in base 10.
- You will learn about the “two’s complement” method to negate a binary number.

Motivation

- We want to be able to represent negative numbers inside the computer.
- We want the usual arithmetic to work with the new representation for negative numbers.

Traditional negative decimals

- In base 10, we traditionally use the '-' symbol to represent a negative number.
- Examples:
  - -345_{10}
  - -7_{10}
- We can invoke arithmetic operations between positive numbers and negative numbers:
  - $(-5_{10}) + 7_{10} = 7 - 5 = 2_{10}$
  - $(-11_{10}) - (-32_{10}) = 32 - 11 = 21_{10}$
- It seems like everything works right.

Drawbacks of the traditional “-”

- Before adding any two numbers, we have to check their signs, and act accordingly:
  - $a + (-b) = a - b$
  - $(-a) + b = b - a$
  - $(-a) + (-b) = -(a + b)$
  - $a + b = a + b$
- The usual subtraction is hard to invoke. (We have to handle borrows from far away bits).
  - We want to only have only the addition operator.
  - To calculate $a - b$ We calculate instead $a + (-b)$.
- We have to keep the “$-$” sign before any signed number. That means keeping one extra symbol before every number.

Subtraction by addition

- We want to solve the following: $932_{10} - 151_{10} =?$
- We add and remove 1000_{10}:
  - $932_{10} - 151_{10} = 932_{10} + (1000_{10} - 151_{10}) - 1000_{10}$
  - $1000_{10} - 151_{10} = 849_{10}$ ("Mechanical" operation).
- We replace the subtraction with the result:
  - $932_{10} - 151_{10} = 932_{10} + 849_{10} - 1000_{10} = 1781_{10} - 1000_{10}$
  - $1781_{10} - 1000_{10} = 781_{10}$ ("Mechanical" operation)
- We conclude that $932_{10} - 151_{10} = 781_{10}$.
  - We only used addition to solve it.
  - All the subtraction operations were mechanical.

Subtraction by addition (Cont.)

- We saw that:
  - $932_{10} - 151_{10} = 781_{10}$
  - $932_{10} + 849_{10} = 1781_{10}$

Subtraction by addition (Cont.)

- We saw that:
  - $932_{10} - 151_{10} = 781_{10}$
  - $932_{10} + 849_{10} = 1781_{10}$

Subtraction by addition (Cont.)

- We saw that:
  - $932_{10} - 151_{10} = 781_{10}$
  - $932_{10} + 849_{10} = 1781_{10}$
- Subtraction by Addition!
- What is the relation between 151_{10} and 849_{10}?
  - $1000_{10} - 151_{10} = 849_{10}$
  - Mechanically: Every digit d is replaced by $(9 - d)$, and finally we add 1 to the result.
  - 849_{10} is virtually the negation of 151_{10}, in the world of 3 decimal digits.
  - We call this method of turning 151_{10} into 849_{10}: The ten's complement.

The ten's complement

- Summary of the method:
  - We want to calculate $637_{10} - 291_{10} = ?$
  - We confine ourselves to the world of 3 decimal digits.
  - We find the ten's complement of 291_{10} by finding $9 - d$ for every digit, and finally adding 1:
    - $291_{10} \rightarrow 708_{10} + 1_{10} = 709_{10}$
  - We add: $637_{10} + 709_{10} = 1346_{10}$, and consider only the lowest 3 digits.
  - We get that $637_{10} - 291_{10} = 346_{10}$

Take a break

- And come back when you are ready for the Binary version of ten's complement.

The two’s complement

- We apply the same method to binary numbers.
- We want to calculate $10110_2 - 111_2 =?$
  - We confine ourselves to the world of 5 bits.
  - We find two’s complement of 111_2 by calculating $(1-b)$ for every bit b (“Flip” every bit), and finally adding 1_2.
    - $111_2 = 00111_2 \rightarrow 11000_2 + 1_2 = 11001_2$
  - We add $10110_2 + 11001_2 = 101111_2$ and consider only the lowest 5 digits.
  - We get that $10110_2 - 111_2 = 01111_2 = 1111_2$

Observations

- Using the two's complement imposes a limit on the amount of bits used.
  - We have to fix the amount of bits used before finding the two's complement of a number.
  - Example: $101_2 \rightarrow 011_2$ in the world of 3 bits, however $101_2 \rightarrow 111011_2$ in the world of 6 bits.
- Adding a number with his complement gives 0.
  - Example: $101_2 + 111011_2 = 1000000_2$

Signed binary numbers

- Let us look at all the binary numbers with 8 bits.
  - Also called **Byte**.
- We call numbers that begin with the bit "1" negative.
- To change the sign of a number we use the two's complement method.
  - Example: The number 10110001_2 is a negative number. $10110001_2 \to 01001110_2 + 1_2 = 01001111_2 = 79_{10}$. Hence 10110001_2 represents -79_{10}.
- We call this representation Signed Binary Numbers of size 8, as opposed to the simple representation that is called Unsigned Binary Numbers of size 8.
- We can add signed numbers "in the usual way", and the sign of the numbers is taken into account automatically!

Examples of signed addition

- Example: $46_{10} - 17_{10} = 29_{10}$
- In binary:
  - $46_{10} = 00101110_2$
  - $17_{10} = 00010001_2$.
    - $00010001_2 \rightarrow 11101110_2 + 1_2 = 11101111_2 = -17_{10}$
  - $46_{10} - 17_{10} = 46_{10} + (-17_{10}) = 00101110_2 + 11101111_2 = 1\textcolor{red}{00011101}_2$
    - We consider only the lowest 8 bits.
  - $00011101_2 = 29_{10}$
- The "+" operator works well with two's complement.

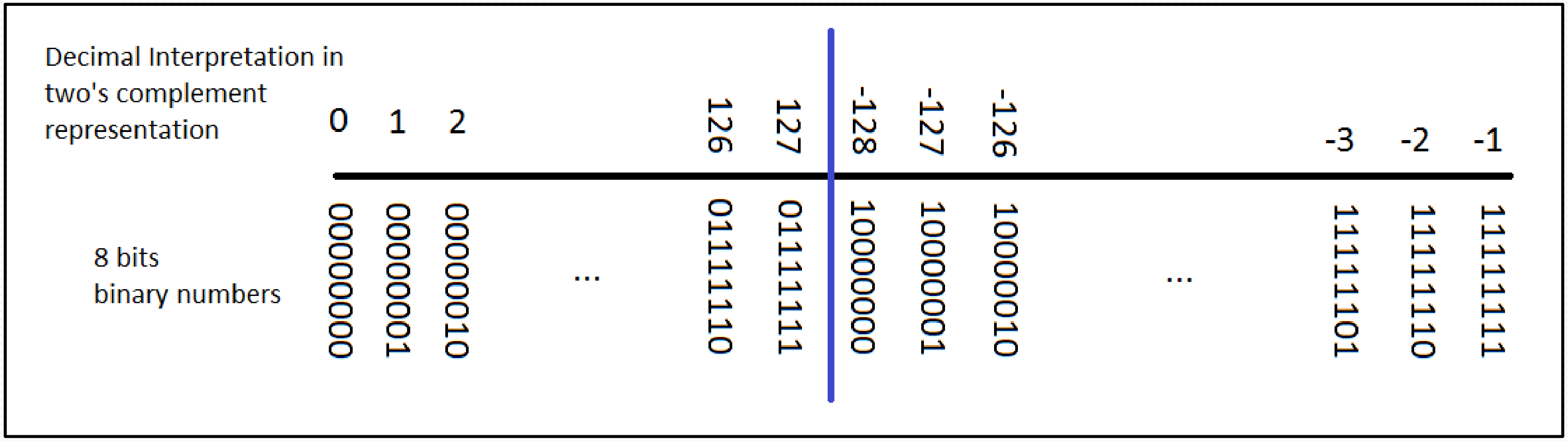
Examples of signed addition (Cont.)

- Example: $-15_{10} - 101_{10} = -116_{10}$
- In binary:
  - $15_{10} = 00001111_2$
    - $-15_{10} = 11110000_2 + 1_2 = 11110001_2$
  - $101_{10} = 01100101_2$
    - $-101_{10} = 10011010_2 + 1_2 = 10011011_2$
  - $-15_{10} - 101_{10} = (-15_{10}) + (-101_{10}) = 11110001_2 + 10011011_2 = 110001100_2$
    - We consider only the lower 8 bits.
- 10001100_2 begins with 1, it is a negative number.
  - $10001100_2 \rightarrow 01110011_2 + 1_2 = 01110100_2 = 116_{10}$

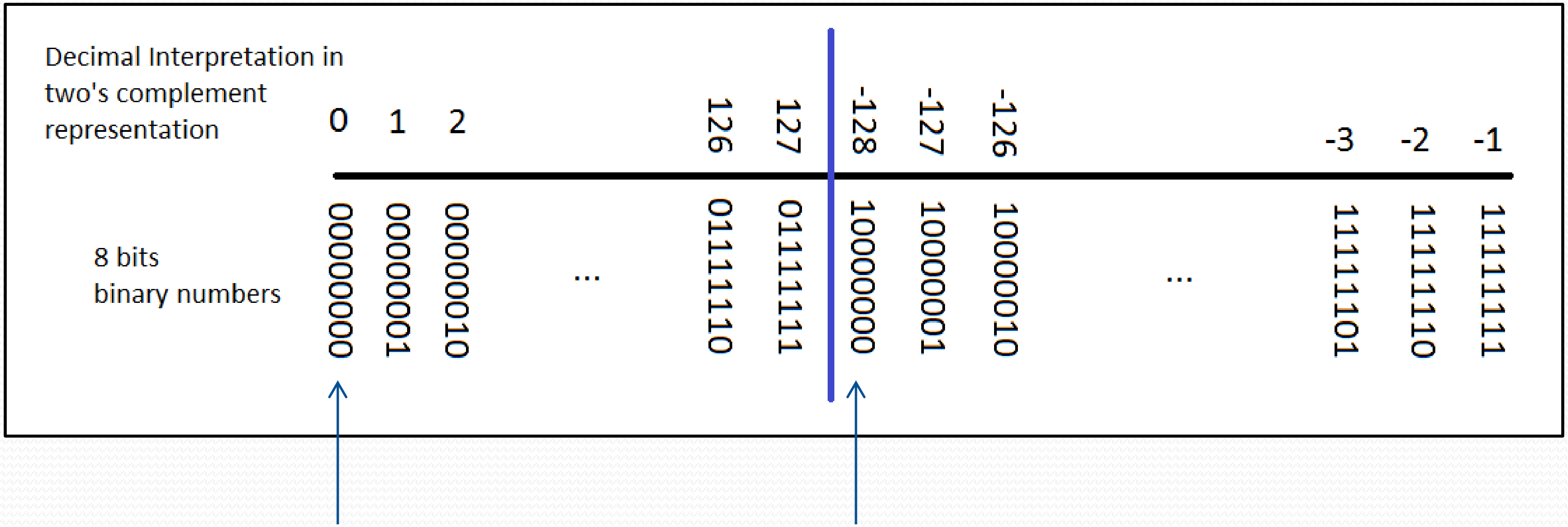
Exceptions

- The two's complement takes positive numbers into negative numbers and vice versa.
  - The highest bit is usually flipped after invoking two's complement.
- Two Exceptions:
  - 0 complements himself.
    - $0_2 \rightarrow 11111111_2 + 1_2 = 1\textcolor{red}{00000000}_2$
    - Begins with 0, therefore it is formally positive.
  - "The most negative number" 10000000_2 complements himself:
    - $10000000_2 \rightarrow 01111111_2 + 1_2 = 10000000_2$
    - Also called the "weird number".

Graphical view



Graphical view



Numbers that complement themselves

Some Philosophy of representation

- You now know about at least two interpretations for every binary number you see.
  - How could you decide which interpretation is the right one?
- The bits don't know what they represent, and they don't care.
  - 10001100_2 could mean -116_{10} in two's complement representation, or 140_{10} in simple representation.
- The meaning of a number is obtained from your thoughts about it.
  - And the actions you perform on it, accordingly.

Some philosophy (Cont.)

- Example:
  - $01001011_2 + 11100111_2 = 100110010_2$
- This could mean that:
  - $75_{10} + 231_{10} = 306_{10}$
    - Because $75_{10} = 01001011_2; 231_{10} = 11100111_2; 100110010_2 = 306_{10}$ in the unsigned binary interpretation.
- This could also mean that:
  - $75_{10} - 25_{10} = 50_{10}$
    - Because $75_{10} = 01001011_2; 11100111_2 = -25_{10}; 00110010_2 = 50_{10}$ in the signed two's complement interpretation.
- Amazingly, both are correct.

Exercises

- Basic ten’s and two’s complement calculations.
  - Use a pen and a paper to solve. Use a calculator to check your results.
- Some more interesting exercises.