

ECE 2020

Number Systems: Rebooted

Instructor: Samuel Talkington

October 8, 2024

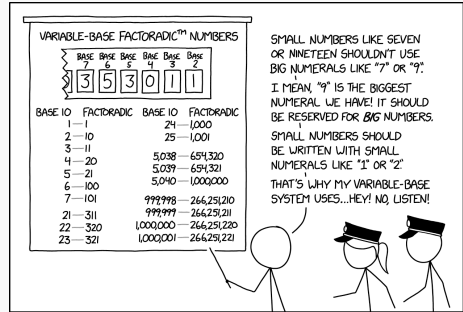
Logistics

- **HW3:** Due Wednesday, October 9th or Friday, October 11th.
- **Exam 1:**
 - Correction opportunity in office hours closes this week.
 - You should have received a message from me on Canvas if you are eligible.
 - If these times don't work for you, feel free to reach out.
- **Exam 2:** October 17th.
- **Midterm survey:** complete for +1 bonus point on your participation grade.

Agenda

Agenda: next 2 weeks

- Circuit timing
- Number systems
- Encoders/decoders
- Multiplexers
- Adders and subtractors



FACTORIAL NUMBERS ARE THE NUMBER SYSTEM THAT SOUNDS MOST LIKE A PRANK BY SOMEONE WHO'S ABOUT TO BE ESCORTED OUT OF THE MATH DEPARTMENT BY SECURITY.

Source: xkcd

Numbers: How do they work?

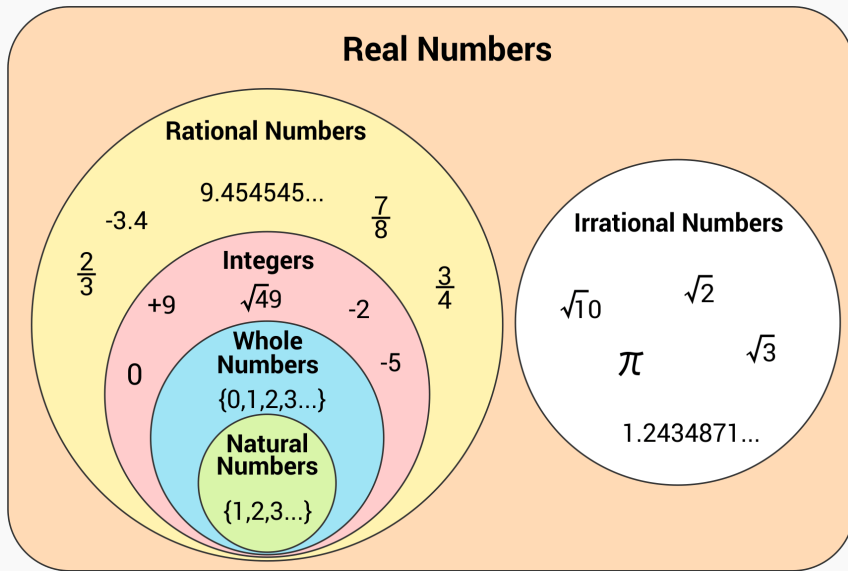


Figure 1: The subsets of the real numbers

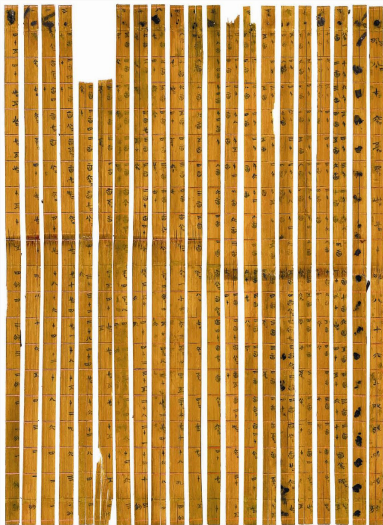


Figure 2: Oldest known base-10 multiplication table, China, c. 305 BC

NUMERALS	1	2	3	4	5	6	7	8	9	10	20	30	40	50	60	70	80	90	100	200	1000
Aśoka		+	6											6							4
Nānā Ghāt	=	+	4	7	2	α	0								1	0	H	H	T		
Nasik	=	≡	+	1	7	7	7	3	α	0					X					7	7
Kṣatrapa	=	≡	+	1	7	7	7	3	α	0					X					7	7
Kuṣana	=	≡	+	1	7	7	7	3	α	0					X					7	7
Gupta	=	≡	+	1	7	7	7	3	α	0					X					7	7
Valhabī	=	≡	+	1	7	7	7	3	α	0					X					7	7
Nepal	=	≡	+	1	7	7	7	3	α	0					X					7	7
Kaliṅga	=	≡	+	1	7	7	7	3	α	0					X					7	7
Vākāṭaka	=	≡	+	1	7	7	7	3	α	0					X					7	7

Figure 3: Evolution of Hindu-Arabic numerals, starting with Edicts of Ashoka, c. 250 BC

𐎶 1	𐎶𐎵 11	𐎶𐎵𐎶 21	𐎶𐎵𐎶𐎵 31	𐎶𐎵𐎶𐎵𐎶 41	𐎶𐎵𐎶𐎵𐎶𐎵 51
𐎶𐎶 2	𐎶𐎶𐎵 12	𐎶𐎶𐎵𐎶 22	𐎶𐎶𐎵𐎶𐎵 32	𐎶𐎶𐎵𐎶𐎵𐎶 42	𐎶𐎶𐎵𐎶𐎵𐎶𐎵 52
𐎶𐎶𐎶 3	𐎶𐎶𐎶𐎵 13	𐎶𐎶𐎶𐎵𐎶 23	𐎶𐎶𐎶𐎵𐎶𐎵 33	𐎶𐎶𐎶𐎵𐎶𐎵𐎶 43	𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 53
𐎶𐎶𐎶𐎶 4	𐎶𐎶𐎶𐎶𐎵 14	𐎶𐎶𐎶𐎶𐎵𐎶 24	𐎶𐎶𐎶𐎶𐎵𐎶𐎵 34	𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 44	𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 54
𐎶𐎶𐎶𐎶𐎶 5	𐎶𐎶𐎶𐎶𐎶𐎵 15	𐎶𐎶𐎶𐎶𐎶𐎵𐎶 25	𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 35	𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 45	𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 55
𐎶𐎶𐎶𐎶𐎶𐎶 6	𐎶𐎶𐎶𐎶𐎶𐎶𐎵 16	𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶 26	𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 36	𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 46	𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 56
𐎶𐎶𐎶𐎶𐎶𐎶𐎶 7	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵 17	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶 27	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 37	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 47	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 57
𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶 8	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵 18	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶 28	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 38	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 48	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 58
𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶 9	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵 19	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶 29	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 39	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 49	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶𐎵 59
𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶 10	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵 20	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶 30	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵 40	𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎶𐎵𐎶𐎵𐎶 50	

The Babylonian cuneiform numerals, c. 2000 BC, were the first positional number system; shockingly, they were **base-60**, or *sexagesimal*. (???)

Bottom line

How we represent numbers is a *choice of human definition*.

Figure 4: Babylonian cuneiform numerals, c. 2000 BC

Classroom discussion

There's an infinite number of real numbers, (“uncountably” infinite), and an infinite number of natural numbers, (“countably” infinite). Yet, we represent everything in terms of *just 10* of the natural numbers: 0, 1, ..., 9.

Think about it for a second:

- How can we store numbers in computers?
- What kind of numbers would fit well into digital logic design?

Positional number systems

Positional number systems i

Question: How exactly do we represent a number?

Answer: We have to agree on the total number of **unique**, or **base** numbers to build our numbers from; the number of such unique numbers is called the *radix*.

$$\text{usual digits} = \underbrace{\{0, 1, 2, \dots, 9\}}_{\text{\#digits}=b,}$$

The total number of unique digits in a number system, b , is called the **radix**.

Positional number systems ii

How? Positional numbers work by exponentiating the radix, multiplying the value of its place, and summing all of these together.

Example:

$$(241)_{10} = (2 \times 10^2) + (4 \times 10^1) + (1 \times 10^0)$$

We can make this more general!

Definition: Base- b number system

A base- b number system with radix $b > 1$ represents any number $x \in \mathbb{R}$ as a string of digits a_i in n “places” $i = 0, 1, \dots, n-1$, where each a_i is one of b possible digits in a digit set \mathcal{D} :

$$a_i \in \mathcal{D} = \{d_1, d_2, \dots, d_b\}.$$

Any real number x can be represented in a base- b system as the following sum:

$$\underline{x} = \underline{(a_{n-1}a_{n-2} \dots a_1a_0)_b} = \sum_{i=0}^{n-1} a_i \times b^i. \quad (1)$$

basis expansion

Base-10 number system

Base-10 numbers are the numbers we all know and love.

Examples:

- $\underbrace{3}_{=a_0} = 3 \times 10^0$

- $\underbrace{53}_{a_1 a_0} = 5 \times 10^1 + 3 \times 10^0$

- $\underbrace{125}_{=a_2 a_1 a_0} = \sum_{i=0}^2 a_i \times 10^i = a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0$

Base-2 (Binary) number system

Base-2 (a.k.a. binary) numbers are the numbers we are all (starting) to know and love.

Examples:


$$\bullet \underbrace{10}_{a_1 a_0} = 1 \times 2^1 + 0 \times 2^0 = (2)_{10}$$

$$\bullet \underbrace{110}_{a_2 a_1 a_0} = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (6)_{10}$$

$$\bullet \underbrace{11010}_{a_4 a_3 a_2 a_1 a_0} = \sum_{i=0}^3 a_i \times 2^i = 2^4 + 2^3 + 0 + 2^1 + 0 = (26)_{10}$$

Base-16 (Hexadecimal) number system

In **base-16** or *hexadecimal* numbers, the set of base digits are:

$$\mathcal{D} = \{0, 1, 2, \dots, 9, A, B, C, D, E, F\},$$


where:

$$(A)_{10} = \underline{10}, \quad (B)_{10} = \underline{11}, \quad (C)_{10} = \underline{12},$$

$$(D)_{10} = \underline{13}, \quad (E)_{10} = \underline{14}, \quad (F)_{10} = \underline{15}.$$

Why care about non-base-10?

- base-2 (binary): Digital logic, all of computing instruction are converted to this.
- base-8: 3-bit information (useful in analysis, prototyping)
- base-16: 4-bit information (**tons** of computer stuff)
 - 32-bit IP addresses are 8 digits
 - 32-bit CPU instructions are 8 digits
- base-60: Deciphering ancient Babylonian Cuneiform tablets (**essential**)

Converting between number systems

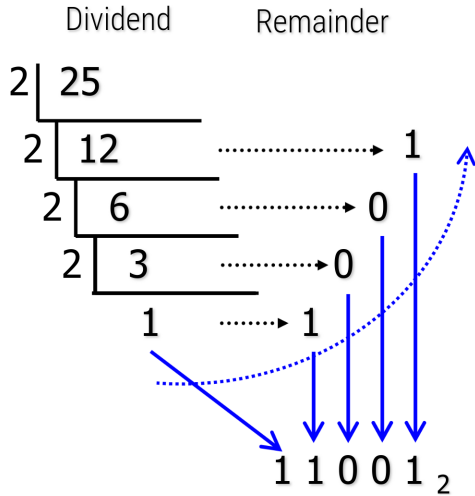


Figure 5: General procedure for converting $(25)_{10}$ to binary.

Converting between number systems i

To **convert** a base- b number to binary (base-2), you can follow these general concepts:

- Convert the number to base-10 using the appropriate number system.
- Divide the decimal number by 2.
- Note the remainder.
- Repeat the previous 2 steps for the quotient till the quotient is zero.
- Write the remainders in reverse order.

Converting between number systems ii

Example: converting a hexadecimal number to binary

How to convert this hexadecimal number to binary?

$$(\underline{2A})_{16} = (?)_2.$$

Solution: For the case of hex→binary, you can individually convert each digit into a 4-bit binary digit.

$$\underline{(2A)}_{16} = \underline{(42)}_{10} = \underbrace{0010}_{=(4)_{10}} \underbrace{1010}_{=(10)_{10}}$$

Examples i

$$(A)_{16} = (10)_{10}$$
$$(B)_{16} = (11)_{10}$$

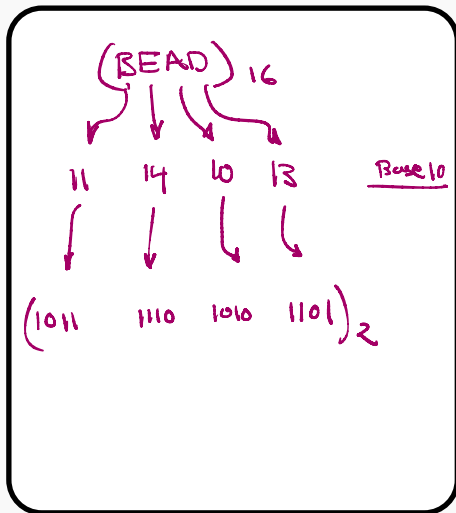
Example

Convert:

$$(BEAD)_{16} = (?)_2$$

Answer:

$$(BEAD)_{16} = (1011111010101101)_2$$



Examples ii

Example

Convert:

$$(10.1011001011)_2 = (?)_{16}$$

Answer:

$$(10.1011001011)_2 = (2.B2C)_{16}$$

Note the trailing zeros

$$\begin{aligned}(10.1011001011)_2 &= (?)_{16} \\ &= \left(2^1 \cdot \left(2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} + 2^{-7} + \dots + 2^{-9} + 2^{-10} \right) \right)_{10} \\ &= (2.B2C)_{16}\end{aligned}$$

Fractional number representations

Fixed-point fractional representation i

Consider the number 5.75 in base-10:

$\underbrace{5}_{\text{whole part}} \underbrace{.}_{\text{decimal point}} \underbrace{75}_{\text{fractional part}}$

Equivalently, in binary, $5.75 = 101.11$:

$\underbrace{101}_{\text{whole part}} \underbrace{.}_{\text{binary point}} \underbrace{11}_{\text{fractional part}}$

Fixed-point fractional representation ii

The reason for this is because:

$$(5)_{10} = (101)_2,$$

$\begin{matrix} 2^2 & 2^0 \\ \downarrow & \downarrow \end{matrix}$

$5.75 = 5 + 0.75$

and

$$\begin{aligned} (0.75)_{10} &= (0.5)_{10} + (0.25)_{10} \\ &= 1 \times 2^{-1} + 1 \times 2^{-2} \end{aligned}$$

$\begin{matrix} \frac{1}{2} & \frac{1}{4} = \frac{1}{2^2} \\ \uparrow & \uparrow \end{matrix}$

We can be put this in a general form:

Fractional number systems

A base b fractional number D with n whole number digits $D_{n-1}, D_{n-2}, \dots, D_0$ and r fractional digits $D_{-1}, D_{-2}, \dots, D_{-r}$, can be written as

$$D = \left(\underbrace{D_{n-1}D_{n-2} \dots D_1D_0}_{n \text{ whole digits}} . \underbrace{D_{-1}D_{-2} \dots D_{-r}}_{r \text{ fractional digits}} \right)_b$$

Handwritten notes: Above the fractional part, indices -1, -2, ..., -r are written. A pink arrow points from the decimal point to the start of the fractional digits.

and can be equivalently represented as

$$D = \sum_{i=-r}^{n-1} D_i b^i$$

$$5.75 = 5 + 0.75$$

$$0.75 = 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

Example

Convert:

$$(100.101)_2 = (?)_{10}$$

Answer:

$$(100.101)_2 = (4.625)_{10}$$

4.?

$$(.101)_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

$$= \frac{1}{2} + \frac{1}{8}$$

$$= \frac{4}{8} + \frac{1}{8}$$

$$= \left(\frac{5}{8}\right)_{10}$$

$$= (4.625)_{10}$$

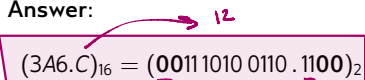
Examples ii

Example

Convert:

$$(3A6.C)_{16} = (?)_2$$

Answer:


$$(3A6.C)_{16} = (\underline{0011} 1010 0110 . \overset{12}{1100})_2$$

Note the trailing and leading zeros above. Equivalently:

$$(3A6.C)_{16} = (111010 0110 . 11)_2$$

$$(3A6.C)_{16} = (?)_2$$

$$3 = (11)_2 = (0011)_2$$

$$(C)_{16} = (\overset{11}{11}00)_2$$

$2^3 2^2 = 8 + 4 = 12$

IMPORTANT: Trailing and Leading Zeros Rule

The rule for where to add 0's is *incredibly important* for correct conversions. For example, consider the conversion: $(22)_{10} = (10110)_2$. If we incorrectly add trailing zeros to convert this to hex, we would get

$$(10110000)_2 = (B0)_{16} = (176)_{10} \neq (22)_{10} \quad (\text{false!})$$

The correct way to convert this to hex is adding **leading zeros** to get

$$(00010110)_2 = (16)_{16} = (22)_{10} \quad (\text{true!})$$

IMPORTANT: Trailing and Leading Zeros Rule

Another example: $(2.5)_{10} = (10.1)_2$. If we want to convert this to hexadecimal, the correct way adding **leading and trailing zeros** around the decimal point:

$$(2.5)_{10} = (\underbrace{0010}_{\text{integer}}.\underbrace{1000}_{\text{fraction}})_2 = (1 \times 2^1) + (1 \times 2^{-1}) = (2.5)_{10} = (2.8)_{16}.$$

If we add trailing 0's to the right both before and after the decimal, we'll incorrectly get

$$(\underbrace{1000}_{\text{integer}}.\underbrace{1000}_{\text{fraction}})_2 = (8.8)_{16} = \underline{(8.5)_{10}} \neq \underline{(2.5)_{10}} \quad (\text{false!})$$

Furthermore, if we add leading zeros on both sides, we'll get

$$(\underbrace{0010}_{\text{integer}}.\underbrace{0001}_{\text{fraction}})_2 = (2.1)_{16} = \underline{(2.0625)_{10}} \neq \underline{(2.5)_{10}} \quad (\text{false!}),$$

Binary addition



Binary addition i

Recall the basic rules:

- $0 + 0 = 0$
- $0 + 1 = 1 + 0 = 1$
- $1 + 1 = 10$

← ↻

$$A, B \in \{0, 1\}$$

$$S = A \oplus B$$

$$\underline{C_{out} = A \cdot B}$$

Binary addition ii

Example

Convert the following base-10 numbers into binary and perform the addition:

$$\begin{array}{r} (190)_{10} \\ + (141)_{10} \\ = (?)_2 \end{array}$$

Answer:

$$(331)_{10}$$

overflow

Conversion:

10111110
+10001101

= 101001011

total # digits

$= \sum_{i=0}^{n-1} D_i \cdot 2^i$

$= 2^0 + 2^1 + 2^3 + 2^6 + 2^8$

$= (331)_{10}$

Example

Convert the following base-10 numbers into binary and perform the addition:

$$\begin{array}{r} (173)_{10} \\ + (44)_{10} \\ \hline = (?)_2 \end{array}$$

Answer:

$$(217)_{10} = (11011001)_2$$



$$(173)_{10} =$$

$$(44)_{10} =$$

$$(10101101)_2$$

$$(00101100)_2$$

$$\hline (11011001)_2$$

How to do subtraction? Need negative number systems.

Negative number representations

Negative numbers i

Negative binary numbers are a system for representing negative numbers as binaries.

→ MSB \equiv leftmost

Signed magnitude binary representation:

- The most significant bit denotes the sign:
 - 0 \rightarrow positive number
 - 1 \rightarrow negative number
- Example:

$$(-25)_{10} = (\textcolor{violet}{1}(25_{10})_2)_2 = (111001)_{2-\text{sgn}-\text{smag}}$$

More examples:

- $(+0)_{10} = (000)_{2\text{-sgn-mag}}$
- $(+1)_{10} = (001)_{2\text{-sgn-mag}}$
- $(+3)_{10} = (011)_{2\text{-sgn-mag}}$
- $(-3)_{10} = (110)_{2\text{-sgn-mag}}$

Why we don't use sign magnitude often:

- Representing zeros is tricky
- Complicated to use in circuits
- Other systems are easier to calculate
- We won't focus on this in this course, but you should be able to recognize them.

One's Complement Negative Number System

The **most significant bit** indicates the sign as in signed magnitude representations, **and**, if a number is negative (i.e., $\text{MSB} = 1$), then **all other bits are complemented**.

One's complement ii

Example

Convert into 1's complement binary representations:

$$(+25)_{10} = (?)_{1\text{comp}}$$

$$(-25)_{10} = (?)_{1\text{comp}}$$

Answer:

$$(+25)_{10} = (011001)_{1\text{comp}}$$

$$(-25)_{10} = (100110)_{1\text{comp}}$$

$$\begin{aligned} & (-25)_{10} \\ &= (1 \underbrace{(+25)_{10}}_2)_2 \\ &= (1 \underbrace{11001}_2)_2 \\ &= (100110)_{1\text{comp}} \end{aligned}$$

One's complement iii

More examples of 1's complement:

- $(+3)_{10} = (011)_{1\text{comp}}$
- $(-3)_{10} = (100)_{1\text{comp}}$
- $(+2)_{10} = (010)_{1\text{comp}}$

$$\begin{array}{r} -3 \\ +2 \\ \hline = (-1)_{10} \end{array}$$

$$\begin{array}{r} (100)_{1\text{comp}} \\ + (010)_{1\text{comp}} \\ \hline (110)_{1\text{comp}} \end{array}$$

MSB=1 \Rightarrow negative

Complement:
 $(01)_2$
 $= -1_{10}$

Problem! Multiple representations for zero:

- $(+0)_{10} = (000)_{1\text{comp}}$
- $(-0)_{10} = (111)_{1\text{comp}}$

Problem! Arithmetic doesn't always work:

$$(0)_{10} + (1)_{10} \stackrel{?}{=} ((111)_{1\text{comp}} + (001)_{1\text{comp}})_{10} = (0)_{10} \dots \text{False!}$$

Two's Complement

The *Two's Complement* representation of binary numbers is formed by taking the One's complement and adding 1, *ignoring any overflow*.

Primary method we will use in this course!

Why 2's complement?

- Arithmetic always works ✓
- Unique representation for zero ✓
- Support an extra negative value over 1's complement. ✓

Procedure for reading a 2's complement number

- Method 1:

- (1) Complement using 2's complement
- (2) Find the value & place a negative sign

- Method 2:

- (1) Convert the value of all bits except the MSB as if it were a positive number
- (2) Let i_{MSB} be the place of the MSB. Subtract $2^{(i_{\text{MSB}})_{10}}$ from the converted number.

2's comp: 1001
 ↑
 8

Two's complement iv

Example

Convert the following 2's complement number to base-10:

$$(101)_{2\text{comp}} = (?)_{10}$$

Answer: $(-3)_{10}$ ✓

$$\begin{array}{l} 2^2 \\ \downarrow \\ (101)_{2\text{comp}} \quad i_{\text{msb}} = 2 \\ \downarrow \\ (1)_{10} \\ 1 - 2^2 = (-3)_{10} \end{array}$$

Two's complement v

Example

Convert the following 2's complement number to base-10:

$$\overbrace{(1010.1100)}^2_{2\text{comp}} = (?)_{10}$$

Answer: $(-5.25)_{10}$

Whole: (negative, MSB=1)

MSB=3

Whole=2

Fractional

$$1 \cdot 2^{-1} + 1 \cdot 2^{-2} = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$$

$$2.75 - 2^3$$

$$= (-5.25)_{10}$$

Arithmetic with 2's complement

Addition, subtraction, multiplication

- Addition: same as before ✓
- Subtraction: $A_{2\text{comp}} - B_{2\text{cmp}}$ ✓
 - 1.) Complement B using 2's complement
 - 2.) Add A and the complemented B together:

$$A - B = A + (-B)$$

- Multiplication: Put it in terms of repeated addition.

$$A \times B = \underbrace{A + A + \dots + A}_{B \text{ times}}$$

(Other methods are available, but likely outside the scope of this class.)

Unsigned: A, B

$$A - B = (A)_{2\text{comp}} + (-B)_{2\text{cp}}$$

Ranges of number systems

Consider an n -bit binary number X . We can represent the following ranges of base 10 numbers with each number system:

$$X = 0001 \Rightarrow 0 \leq (X)_{10} \leq 15$$

- Unsigned:

$$\underline{0 \leq (X)_{10} \leq 2^n - 1}$$

- 1's complement:

$$\underline{-2^{n-1} - 1 \leq (X)_{10} \leq 2^{n-1} - 1}$$

- 2's complement:

$$\underline{-2^{n-1} \leq (X)_{10} \leq 2^{n-1} - 1}$$

Overflow

Definition: Overflow

If the result of an arithmetic operation exceeds the **range** of a number system, we say that **overflow** has occurred.

WARNING

Overflow can **break arithmetic**. Example: $(5)_{10} + (6)_{10} = (?)_{2\text{cmp}}$

$$\begin{array}{r} (0101)_{2\text{cmp}} \\ + (0110)_{2\text{cmp}} \\ \hline = (1011)_{2\text{cmp}} = (-5)_{10} \dots \text{false! error from overflow.} \end{array}$$

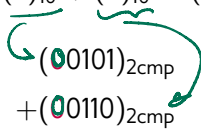
Sign extensions

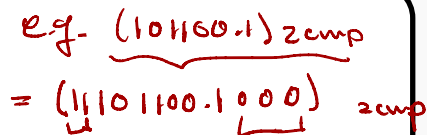
Sign extension: How to fix overflow

Sign extension is a procedure to modify a binary number without changing the value, to remove the risk of overflow.

How to fix overflow

Example: $(5)_{10} + (6)_{10} = (?)_{2\text{cmp}}$


$$\begin{array}{r} (00101)_{2\text{cmp}} \\ + (00110)_{2\text{cmp}} \\ \hline = (01011)_{2\text{cmp}} = (+11)_{10} \dots \text{correct! fixed by sign extension.} \end{array}$$


$$\begin{array}{l} \text{e.g. } (101100.1)_{2\text{cmp}} \\ = (\underbrace{11}_{\text{left}} \underbrace{101100.1}_{\text{middle}} \underbrace{000}_{\text{right}})_{2\text{cmp}} \end{array}$$

Overflow example

Overflow example

Perform this addition, note the overflow, and fix it with sign extension

$$\begin{array}{r} (-3)_{10} \\ + (-6)_{10} \\ \hline = (?)_2 \end{array}$$

Answer:

$$(110111)_{2\text{cmp}}$$