

Computer Architecture

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
(csdsp@bristol.ac.uk)

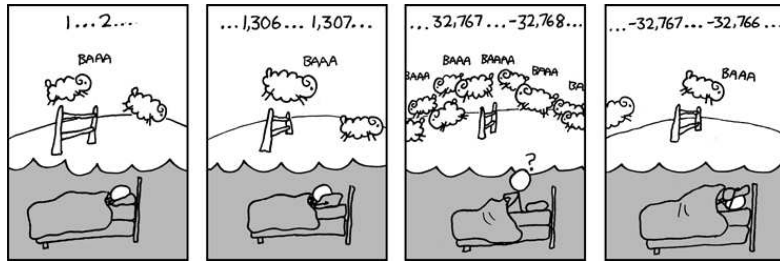
September 17, 2026

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
 - ▶ the associated notes page may be pre-populated with extra, written explanation of material covered in lecture(s), plus
 - ▶ anything with a “grey’ed out” header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:



<https://xkcd.com/571>

© Daniel Page (csdsp@bristol.ac.uk)
Computer Architecture



University of
BRISTOL

Notes:

- **Claim:** at least conceptually we could say that

$$123 \equiv \langle 3, 2, 1 \rangle,$$

i.e., the decimal literal 123 is basically just a sequence of digits.

► Question: given

- a **bit** is a single binary digit, i.e., 0 or 1,
- a **byte** is an 8-element sequence of bits, and
- a **word** is a w -element sequence of bits

and so, e.g.,

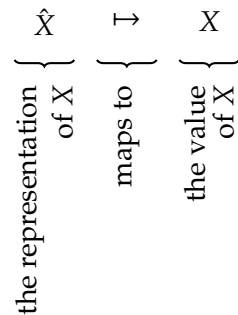
$$01111011 \equiv \langle 1, 1, 0, 1, 1, 1, 1, 0 \rangle,$$

what do these things *mean* ... what do they *represent*?

► Answer: anything *we* decide they do!

Notes:

► Concept:



i.e., we need

1. a concrete representation that we can write down, plus
2. a mapping that yields the correct value *and* is consistent (in both directions).

Notes:

► Agenda:

1. useful properties of bit sequences,
2. positional number systems $\leadsto$ standard integer representations.

Notes:

Part 1: useful properties of bit sequences

Definition

A given literal, say

$$X = 1111011,$$

can be interpreted in *two* ways:

1. A **little-endian** ordering is where we read bits in a literal from right-to-left, i.e.,

$$\hat{X}_{LE} = \langle X_0, X_1, X_2, X_3, X_4, X_5, X_6 \rangle = \langle 1, 1, 0, 1, 1, 1, 1 \rangle.$$

2. A **big-endian** ordering is where we read bits in a literal from left-to-right, i.e.,

$$\hat{X}_{BE} = \langle X_6, X_5, X_4, X_3, X_2, X_1, X_0 \rangle = \langle 1, 1, 1, 1, 0, 1, 1 \rangle.$$

Notes:

Definition

Given a bit-sequence X , the **Least-Significant Bit (LSB)** and **Most-Significant Bit (MSB)** are those in X which have the least and most significance, i.e., have the least and most impact on, the value it represents.

Part 1: useful properties of bit sequences

Definition

Following the idea of vectorial Boolean function, given an n -element bit-sequence X , and an m -element bit-sequence Y we can clearly

1. overload $\oslash \in \{\neg\}$, i.e., write

$$R = \oslash X,$$

to mean

$$R_i = \oslash X_i$$

for $0 \leq i < n$,

2. overload $\ominus \in \{\wedge, \vee, \oplus\}$, i.e., write

$$R = X \ominus Y,$$

to mean

$$R_i = X_i \ominus Y_i$$

for $0 \leq i < n = m$, where if $n \neq m$, we pad either X or Y with 0 until the $n = m$.

Notes:

Part 1: useful properties of bit sequences

Definition

Following the idea of vectorial Boolean function, given an n -element bit-sequence X , and an m -element bit-sequence Y we can clearly

1. overload $\oslash \in \{\neg\}$, i.e., write

$$R = \oslash X,$$

to mean

$$R_i = \oslash X_i$$

for $0 \leq i < n$,

2. overload $\ominus \in \{\wedge, \vee, \oplus\}$, i.e., write

$$R = X \ominus Y,$$

to mean

$$R_i = X_i \ominus Y_i$$

for $0 \leq i < n = m$, where if $n \neq m$, we pad either X or Y with 0 until the $n = m$.

Notes:

► **Example:** C **bit-wise** (computational) operators, where, e.g., $\& \equiv \wedge$

$$\begin{aligned} \mathbf{x} &= \mathbf{0x13} \leftarrow \langle 1, 1, 0, 0, 1, 0, 0, 0 \rangle \\ \mathbf{y} &= \mathbf{0x59} \leftarrow \langle 1, 0, 0, 1, 1, 0, 1, 0 \rangle \\ \mathbf{r} = \mathbf{x} \& \mathbf{y} &= \mathbf{0x11} \leftarrow \langle 1 \wedge 1, 1 \wedge 0, 0 \wedge 0, 0 \wedge 1, 1 \wedge 1, 0 \wedge 0, 0 \wedge 1, 0 \wedge 0 \rangle \\ &= \langle 1, 0, 0, 0, 1, 0, 0, 0 \rangle \end{aligned}$$

Part 1: useful properties of bit sequences

Definition

Given two n -bit sequences X and Y , we can define some important properties named after Richard Hamming, a researcher at Bell Labs:

- ▶ The **Hamming weight** of X is the number of bits in X that are equal to 1, i.e., the number of times $X_i = 1$. This can be expressed as

$$\text{HW}(X) = \sum_{i=0}^{n-1} X_i.$$

- ▶ The **Hamming distance** between X and Y is the number of bits in X that differ from the corresponding bit in Y , i.e., the number of times $X_i \neq Y_i$. This can be expressed as

$$\text{HD}(X, Y) = \sum_{i=0}^{n-1} X_i \oplus Y_i = \text{HW}(X \oplus Y).$$

Note that both quantities naturally generalise to non-binary sequences.

Notes:

Part 1: useful properties of bit sequences

Definition

Given two n -bit sequences X and Y , we can define some important properties named after Richard Hamming, a researcher at Bell Labs:

- ▶ The **Hamming weight** of X is the number of bits in X that are equal to 1, i.e., the number of times $X_i = 1$. This can be expressed as

$$\text{HW}(X) = \sum_{i=0}^{n-1} X_i.$$

- ▶ The **Hamming distance** between X and Y is the number of bits in X that differ from the corresponding bit in Y , i.e., the number of times $X_i \neq Y_i$. This can be expressed as

$$\text{HD}(X, Y) = \sum_{i=0}^{n-1} X_i \oplus Y_i = \text{HW}(X \oplus Y).$$

Note that both quantities naturally generalise to non-binary sequences.

Notes:

- ▶ **Example:** given $X = \langle 1, 0, 0, 1 \rangle$ and $Y = \langle 0, 1, 1, 1 \rangle$ we find that

$$\text{HW}(X) = \sum_{i=0}^{n-1} X_i = 1 + 0 + 0 + 1 = 2$$

$$\text{HD}(X, Y) = \sum_{i=0}^{n-1} X_i \oplus Y_i = (1 \oplus 0) + (0 \oplus 1) + (0 \oplus 1) + (1 \oplus 1) = 1 + 1 + 1 + 0 = 3$$

Part 2: positional number systems $\leadsto$ standard integer representations (1)

- **Concept:** a **positional number system** expresses the value of a number x using a base- b (or radix- b) expansion, i.e.,

$$\hat{x} = \langle \hat{x}_0, \hat{x}_1, \dots, \hat{x}_{n-1} \rangle$$

$$\mapsto x$$

$$= \pm \sum_{i=0}^{n-1} \hat{x}_i \cdot b^i$$

where each $\hat{x}_i$

- is one of n digits taken from the digit set $X = \{0, 1, \dots, b-1\}$,
- is “weighted” by some power of of the base b .

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (1)

- **Beware!**
 - for $b > 10$ we can't express $\hat{x}_i$ using a single Arabic numeral,
 - for $b = 16$, for example, we use letters instead:

A	$\mapsto$	10
B	$\mapsto$	11
C	$\mapsto$	12
D	$\mapsto$	13
E	$\mapsto$	14
F	$\mapsto$	15

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (2)

Example

Consider an example where we

1. set $b = 10$, i.e., deal with **decimal** numbers, and
2. have $\hat{x}_i \in X = \{0, 1, \dots, 10 - 1 = 9\}$.

This means we can write

$$\begin{aligned}\hat{x} = 123 &= \langle 3, 2, 1 \rangle_{(10)} \\ &\mapsto x \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 10^i \\ &= 3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2 \\ &= 3 \cdot 1 + 2 \cdot 10 + 1 \cdot 100 \\ &= 123_{(10)}\end{aligned}$$

i.e., represent the value “one hundred and twenty three” in a variety of ways using different bases.

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (2)

Example

Consider an example where we

1. set $b = 2$, i.e., deal with **binary** numbers, and
2. have $\hat{x}_i \in X = \{0, 2 - 1 = 1\}$.

This means we can write

$$\begin{aligned}\hat{x} = 1111011 &= \langle 1, 1, 0, 1, 1, 1, 1 \rangle_{(2)} \\ &\mapsto x \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i \\ &= 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 \\ &= 1 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 + 1 \cdot 32 + 1 \cdot 64 \\ &= 123_{(10)}\end{aligned}$$

i.e., represent the value “one hundred and twenty three” in a variety of ways using different bases.

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (2)

Example

Consider an example where we

1. set $b = 8$, i.e., deal with **octal** numbers, and
2. have $\hat{x}_i \in X = \{0, 1, \dots, 8 - 1 = 7\}$.

This means we can write

$$\begin{aligned}\hat{x} = 173 &= \langle 3, 7, 1 \rangle_{(8)} \\ &\mapsto x \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 8^i \\ &= 3 \cdot 8^0 + 7 \cdot 8^1 + 1 \cdot 8^2 \\ &= 3 \cdot 1 + 7 \cdot 8 + 1 \cdot 64 \\ &= 123_{(10)}\end{aligned}$$

i.e., represent the value “one hundred and twenty three” in a variety of ways using different bases.

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (2)

Example

Consider an example where we

1. set $b = 16$, i.e., deal with **hexadecimal** numbers, and
2. have $\hat{x}_i \in X = \{0, 1, \dots, 16 - 1 = 15\}$.

This means we can write

$$\begin{aligned}\hat{x} = 7B &= \langle B, 7 \rangle_{(16)} \\ &\mapsto x \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 16^i \\ &= 11 \cdot 16^0 + 7 \cdot 16^1 \\ &= 11 \cdot 1 + 7 \cdot 16 \\ &= 123_{(10)}\end{aligned}$$

i.e., represent the value “one hundred and twenty three” in a variety of ways using different bases.

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (3)

► **Problem:** we want to represent elements of $\mathbb{Z}$, *but*

1. it's an infinite set, and
2. so far we've ignored the issue of sign.

► **Solution:** in C, for example, we get

$$\begin{aligned}\text{unsigned char} &\simeq \text{uint8_t} \mapsto \{0, \dots, +2^8 - 1\} \\ \text{char} &\simeq \text{int8_t} \mapsto \{-2^7, \dots, 0, \dots, +2^7 - 1\}\end{aligned}$$

but why *these*, and how do they work?

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (4)

Unsigned

Definition

An unsigned integer can be represented in n bits by using the natural binary expansion. That is, we have

$$\hat{x} = \langle \hat{x}_0, \hat{x}_1, \dots, \hat{x}_{n-1} \rangle$$

$$\mapsto x$$

$$= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i$$

for $\hat{x}_i \in \{0, 1\}$, which yields

$$0 \leq x \leq 2^n - 1.$$

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (5)

Unsigned

Example ($n = 8$)				
11111111	$\mapsto$	$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+255_{(10)}$
$\vdots$				$\vdots$
10000101	$\mapsto$	$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+133_{(10)}$
$\vdots$				$\vdots$
10000000	$\mapsto$	$1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	$+128_{(10)}$
01111111	$\mapsto$	$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+127_{(10)}$
$\vdots$				$\vdots$
01111011	$\mapsto$	$0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+123_{(10)}$
$\vdots$				$\vdots$
00000001	$\mapsto$	$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+1_{(10)}$
00000000	$\mapsto$	$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	$+0_{(10)}$

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (6)

Unsigned

- **Fact:**
 - each hexadecimal digit $\hat{x}_i \in \{0, 1, \dots, 15\}$,
 - four bits gives $2^4 = 16$ possible combinations, so
 - each hexadecimal digit can be thought of as a short-hand for four binary digits.
- **Example:** we can perform the following translation steps

$$\begin{aligned}
 8AC &= \langle \quad C, \quad A, \quad 8, \quad \rangle_{(16)} \\
 &= \langle \langle 0, 0, 1, 1 \rangle_{(2)}, \langle 0, 1, 0, 1 \rangle_{(2)}, \langle 0, 0, 0, 1 \rangle_{(2)} \rangle_{(16)} \\
 &= \langle \quad 0, 0, 1, 1, \quad 0, 1, 0, 1, \quad 0, 0, 0, 1 \quad \rangle_{(16)} \\
 &\mapsto 2220_{(10)}
 \end{aligned}$$

such that in C, for example,

$$\underbrace{04254}_{b=8} \equiv \underbrace{2220}_{b=10} \equiv \underbrace{0x8AC}_{b=16} \mapsto 2220_{(10)}.$$

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (7)

Unsigned

- **Fact:** left-shift (resp. right-shift) of some x by y digits is equivalent to multiplication (resp. division) by b^y .
- **Example:** taking $b = 2$ we find that

$$\begin{aligned} x \times 2^y &= \left(\sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i \right) \times 2^y \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i \times 2^y \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^{i+y} \\ &\equiv x \ll y \end{aligned}$$

and

$$\begin{aligned} x/2^y &= \left(\sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i \right) / 2^y \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^i / 2^y \\ &= \sum_{i=0}^{n-1} \hat{x}_i \cdot 2^{i-y} \\ &\equiv x \gg y \end{aligned}$$

such that in C, for example,

$$\begin{aligned} 0x8AC \ll 2 &\mapsto 2220_{(10)} \times 2^2 = 8880_{(10)} \mapsto 0x22B0 \\ 0x8AC \gg 2 &\mapsto 2220_{(10)} / 2^2 = 555_{(10)} \mapsto 0x22B \end{aligned}$$

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (8)

Unsigned

- **Problem:** set the i -th bit of some x , i.e., x_i , to 1.
- **Solution:** compute

$$x \vee (1 \ll i).$$

Example

If $x = 0011_{(2)}$ and $i = 2$ then we compute

$$\begin{array}{rcl} x & \vee & (1 \ll i) \\ 0011_{(2)} & \vee & (1 \ll 2) \\ 0011_{(2)} & \vee & 0100_{(2)} \\ 0111_{(2)} & & \end{array}$$

meaning initially $x_2 = 0$, then we changed it so $x_2 = 1$.

Notes:

► **Problem:** set the i -th bit of some x , i.e., x_i , to 0.

► **Solution:** compute

$$x \wedge \neg(1 \ll i).$$

Example

If $x = 0111_{(2)}$ and $m = 2$ then we compute

$$\begin{array}{rcll} x & \wedge & \neg & (\quad 1 \quad \ll \quad i \quad) \\ 0111_{(2)} & \wedge & \neg & (\quad 1 \quad \ll \quad 2 \quad) \\ 0111_{(2)} & \wedge & \neg & (\quad 0100_{(2)} \quad) \\ 0111_{(2)} & \wedge & & 1011_{(2)} \\ 0011_{(2)} & & & \end{array}$$

meaning initially $x_2 = 1$, then we changed it so $x_2 = 0$.

Notes:

► **Problem:** extract the i -th bit of some x , i.e., x_i .

► **Solution:** compute

$$(x \gg i) \wedge 1.$$

Example

If $x = 0011_{(2)}$, then

1. if $i = 2$ we compute

$$\begin{array}{rcll} (\quad x \quad \gg \quad i \quad) & \wedge & 1 \\ (\quad 0011_{(2)} \quad \gg \quad 2 \quad) & \wedge & 1 \\ (\quad 0000_{(2)} \quad) & \wedge & 1 \\ 0000_{(2)} & & \end{array}$$

meaning $x_2 = 0$, or

2. if $i = 0$ we compute

$$\begin{array}{rcll} (\quad x \quad \gg \quad i \quad) & \wedge & 1 \\ (\quad 0011_{(2)} \quad \gg \quad 0 \quad) & \wedge & 1 \\ (\quad 0011_{(2)} \quad) & \wedge & 1 \\ 0001_{(2)} & & \end{array}$$

meaning $x_0 = 1$.

Notes:

- **Problem:** extract an m -bit sub-word (i.e., m contiguous bits) starting at the i -th bit of some x .
- **Solution:** compute

$$(x \gg i) \wedge ((1 \ll m) - 1).$$

Example

If $x = 1011_{(2)}$, $m = 2$ and $i = 1$ then we want to extract the sub-word $\langle x_1, x_2 \rangle$

$$\begin{array}{rclcl} (x & \gg & i) & \wedge & ((1 \ll m) - 1) \\ (1011_{(2)} & \gg & 1) & \wedge & ((1 \ll 2) - 1) \\ (0101_{(2)} & &) & \wedge & (0100_{(2)} - 1) \\ (0101_{(2)} & &) & \wedge & (0011_{(2)} - 1) \\ (0001_{(2)} & &) & & \end{array}$$

meaning $\langle x_1, x_2 \rangle = \langle 1, 0 \rangle$ as expected.

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (12)

Signed, sign-magnitude

Definition

A signed integer can be represented in n bits by using the **sign-magnitude** approach; 1 bit is reserved for the sign (0 means positive, 1 means negative) and $n - 1$ for the magnitude. That is, we have

$$\hat{x} = \langle \hat{x}_0, \hat{x}_1, \dots, \hat{x}_{n-1} \rangle$$

$$\mapsto x$$

$$= (-1)^{\hat{x}_{n-1}} \cdot \sum_{i=0}^{n-2} \hat{x}_i \cdot 2^i$$

for $\hat{x}_i \in \{0, 1\}$, which yields

$$-2^{n-1} + 1 \leq x \leq +2^{n-1} - 1.$$

Note that there are two representations of zero (i.e., +0 and -0).

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (13)

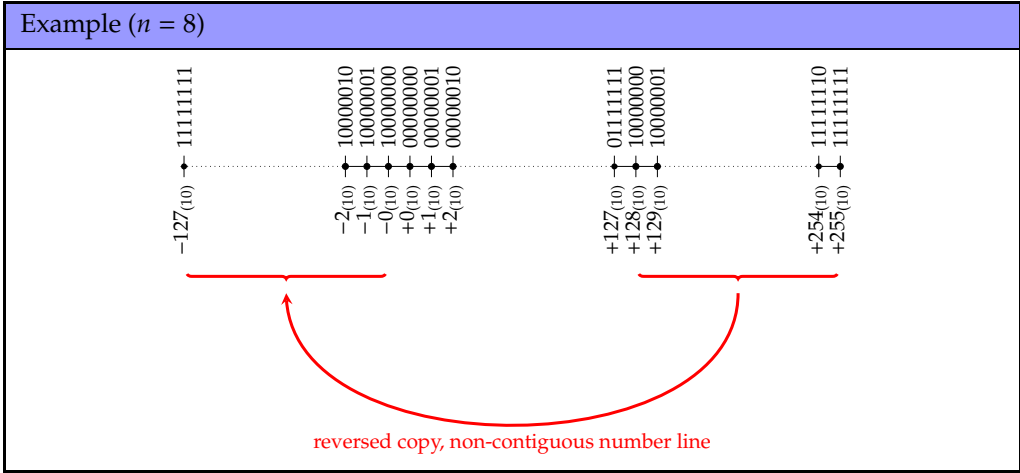
Signed, sign-magnitude

Example ($n = 8$)														
01111111	$\mapsto$	$(-1)^0$	$\cdot$	(	$1 \cdot 2^6$	$+ 1 \cdot 2^5$	$+ 1 \cdot 2^4$	$+ 1 \cdot 2^3$	$+ 1 \cdot 2^2$	$+ 1 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$+127_{(10)}$
$\vdots$														$\vdots$
01111011	$\mapsto$	$(-1)^0$	$\cdot$	(	$1 \cdot 2^6$	$+ 1 \cdot 2^5$	$+ 1 \cdot 2^4$	$+ 1 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 1 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$+123_{(10)}$
$\vdots$														$\vdots$
00000001	$\mapsto$	$(-1)^0$	$\cdot$	(	$0 \cdot 2^6$	$+ 0 \cdot 2^5$	$+ 0 \cdot 2^4$	$+ 0 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 0 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$+1_{(10)}$
00000000	$\mapsto$	$(-1)^0$	$\cdot$	(	$0 \cdot 2^6$	$+ 0 \cdot 2^5$	$+ 0 \cdot 2^4$	$+ 0 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 0 \cdot 2^1$	$+ 0 \cdot 2^0$	)	=	$+0_{(10)}$
10000000	$\mapsto$	$(-1)^1$	$\cdot$	(	$0 \cdot 2^6$	$+ 0 \cdot 2^5$	$+ 0 \cdot 2^4$	$+ 0 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 0 \cdot 2^1$	$+ 0 \cdot 2^0$	)	=	$-0_{(10)}$
10000001	$\mapsto$	$(-1)^1$	$\cdot$	(	$0 \cdot 2^6$	$+ 0 \cdot 2^5$	$+ 0 \cdot 2^4$	$+ 0 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 0 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$-1_{(10)}$
$\vdots$														$\vdots$
11111011	$\mapsto$	$(-1)^1$	$\cdot$	(	$1 \cdot 2^6$	$+ 1 \cdot 2^5$	$+ 1 \cdot 2^4$	$+ 1 \cdot 2^3$	$+ 0 \cdot 2^2$	$+ 1 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$-123_{(10)}$
$\vdots$														$\vdots$
11111111	$\mapsto$	$(-1)^1$	$\cdot$	(	$1 \cdot 2^6$	$+ 1 \cdot 2^5$	$+ 1 \cdot 2^4$	$+ 1 \cdot 2^3$	$+ 1 \cdot 2^2$	$+ 1 \cdot 2^1$	$+ 1 \cdot 2^0$	)	=	$-127_{(10)}$

Notes:

Part 2: positional number systems $\leadsto$ standard integer representations (14)

Signed, sign-magnitude



Notes:

Definition

A signed integer can be represented in n bits by using the **two's-complement** approach; the basic idea is to weight the $(n - 1)$ -th bit using -2^{n-1} rather than $+2^{n-1}$, and all other bits as normal. That is, we have

$$\begin{aligned} \hat{x} &= \langle \hat{x}_0, \hat{x}_1, \dots, \hat{x}_{n-1} \rangle \\ \mapsto x &= \hat{x}_{n-1} \cdot -2^{n-1} + \sum_{i=0}^{n-2} \hat{x}_i \cdot 2^i \end{aligned}$$

for $\hat{x}_i \in \{0, 1\}$, which yields

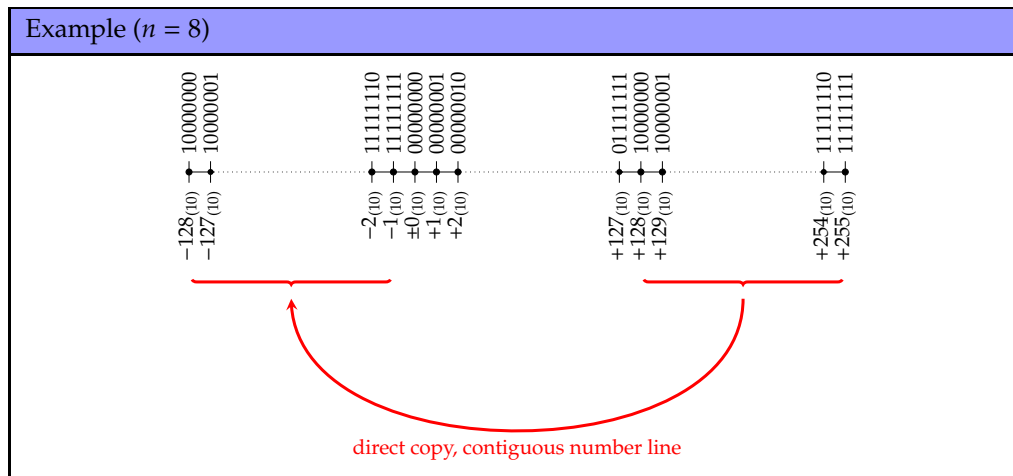
$$-2^{n-1} \leq x \leq +2^{n-1} - 1.$$

Notes:

Example ($n = 8$)

01111111	$\mapsto$	$0 \cdot -2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+127_{(10)}$
$\vdots$				$\vdots$
01111011	$\mapsto$	$0 \cdot -2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+123_{(10)}$
$\vdots$				$\vdots$
00000001	$\mapsto$	$0 \cdot -2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	$+1_{(10)}$
00000000	$\mapsto$	$0 \cdot -2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	$+0_{(10)}$
11111111	$\mapsto$	$1 \cdot -2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	$=$	$-1_{(10)}$
$\vdots$				$\vdots$
10000101	$\mapsto$	$1 \cdot -2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	$=$	$-123_{(10)}$
$\vdots$				$\vdots$
10000000	$\mapsto$	$1 \cdot -2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	$=$	$-128_{(10)}$

Notes:



Notes:

Conclusions

► Take away points:

1. We control what bit sequences mean: we can interpret an instance of the C char data-type as
 - a signed 8-bit integer, *or*
 - a generic object which can take one of 2^8 states,and, as a result, can represent *anything*, e.g.,
 - a pixel within an image,
 - a character within a document,
 - a number within a matrix,
 - ...
2. Beyond this, knowing about various standard representations is important and useful in a general sense.

Notes:

Additional Reading

- ▶ [Wikipedia: Numeral system](#). URL: https://en.wikipedia.org/wiki/Numeral_system.
- ▶ D. Page. “Chapter 1: Mathematical preliminaries”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009.
- ▶ B. Parhami. “Part 1: Number representation”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000.
- ▶ W. Stallings. “Chapter 9: Number systems”. In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013.
- ▶ A.S. Tanenbaum and T. Austin. “Appendix A: Binary numbers”. In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012.

Notes:

References

- [1] [Wikipedia: Numeral system](#). URL: https://en.wikipedia.org/wiki/Numeral_system (see p. 69).
- [2] D. Page. “Chapter 1: Mathematical preliminaries”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009 (see p. 69).
- [3] B. Parhami. “Part 1: Number representation”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000 (see p. 69).
- [4] W. Stallings. “Chapter 9: Number systems”. In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013 (see p. 69).
- [5] A.S. Tanenbaum and T. Austin. “Appendix A: Binary numbers”. In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 69).

Notes: