EGCP 180: -01/02 – Digital Logic and Computer Structures

Spring 2024

Lecture 2: Arithmetic

Rakesh Mahto, Ph.D.
Office*: E 314, California State University, Fullerton*
Office Hour*: Monday and Wednesday 2:00 - 3:30 pm*
Or by appointment
Office Hour Zoom Meeting ID*: 891 2907 5346*
Email*: ramahto@fullerton.edu*
Phone No*: 657-278-7274*

Arithmetic Operations

Arithmetic Operations

- Numbers in base “r” follow the same rules as for decimal numbers; $(r > 1)$
- When using another base be sure to use only the “r” allowable digits; $(r > 1: 0, 1, \ldots, r-1)$

Addition

The addition table for binary numbers is

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 and carry 1 to next column

Carrying 1 to a column
is equivalent to adding 1 to that column.

Add 13_{10} and 11_{10} in binary.

$$
\begin{array}{rl}
 & 1111 \longleftarrow \text{carries} \\
13_{10} = & 1101 \\
11_{10} = & \underline{1011} \\
 & 11000 = 24_{10}
\end{array}
$$

Binary Addition:

1) can only use a 0 or a 1
2) Carry, use with the next significant position higher

Subtraction (a)

The subtraction table for binary numbers is

$0 - 0 = 0$

$0 - 1 = 1$ and borrow 1 from the next column

$1 - 0 = 1$

$1 - 1 = 0$

Borrowing 1 from a column
is equivalent to subtracting 1 from that column.

Subtraction (b)

EXAMPLES OF BINARY SUBTRACTION:

$$
\text{(a)}\quad
\begin{array}{r l}
1 \longleftarrow & \text{(indicates} \\
11101 & \text{a borrow} \\
-10011 & \text{from the} \\
\hline
1010 & \text{3rd column)}
\end{array}
\qquad\qquad
\text{(b)}\quad
\begin{array}{r l}
1111 \longleftarrow & \text{borrows} \\
10000 & \\
-\quad 11 & \\
\hline
1101 &
\end{array}
$$

$$
\text{(c)}\quad
\begin{array}{r l}
111 \longleftarrow & \text{borrows} \\
111001 & \\
-\quad 1011 & \\
\hline
101110 &
\end{array}
$$

Subtraction (c)

A detailed analysis of the borrowing process for this example, indicating first a borrow of 1 from column 1 and then a borrow of 1 from column 2, is as follows:

$$
\begin{aligned}
205 - 18 &= [2 \times 10^2 + 0 \times 10^1 + 5 \times 10^0] \\
&\quad - [\phantom{2 \times 10^2 + {}} 1 \times 10^1 + 8 \times 10^0]
\end{aligned}
$$

note borrow from column 1

$$
\begin{aligned}
&= [2 \times 10^2 + (0 - 1) \times 10^1 + (10 + 5) \times 10^0] \\
&\quad - [\phantom{2 \times 10^2 + (0 - {}} 1 \times 10^1 + \phantom{(10 + 5) \times{}} 8 \times 10^0]
\end{aligned}
$$

note borrow from column 2

$$
\begin{aligned}
&= [(2 - 1) \times 10^2 + (10 + 0 - 1) \times 10^1 + 15 \times 10^0] \\
&\quad - [\phantom{(2 - 1) \times 10^2 + (10 + 0 - {}} 1 \times 10^1 + 8 \times 10^0] \\
&= [1 \times 10^2 + 8 \times 10^1 + 7 \times 10^0] = 187
\end{aligned}
$$

Multiplication (a)

The multiplication table for binary numbers is

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Multiplication (b)

The following example illustrates multiplication of 13_{10} by 11_{10} in binary:

$$
\begin{array}{r}
1101 \\
1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
10001111
\end{array}
= 143_{10}
$$

Multiplication (c)

When doing binary multiplication, a common way to avoid carries greater than 1 is to add in the partial products one at a time as illustrated by the following example:

1111	multiplicand
1101	multiplier
1111	1st partial product
0000	2nd partial product
(01111)	sum of first two partial products
1111	3rd partial product
(1001011)	sum after adding 3rd partial product
1111	4th partial product
11000011	final product (sum after adding 4th partial product)

Binary Division

Binary division is similar to decimal division, except it is much easier because the only two possible quotient digits are 0 and 1.

We start division by comparing the divisor with the upper bits of the dividend.

If we cannot subtract without getting a negative result, we move one place to the right and try again.

If we can subtract, we place a 1 for the quotient above the number we subtracted from and append the next dividend bit to the end of the difference and repeat this process with this modified difference until we run out of bits in the dividend.

Binary Division

The following example illustrates division of 145_{10} by 11_{10} in binary:

```
       1101
     ________
1011 | 10010001
       1011
       ----
        1110
        1011
        ----
         1101
         1011
         ----
           10
```

The quotient is 1101 with a remainder of 10.

Complements

Complements

Allows us to represent negative numbers and we can do subtraction using addition.

Two types for each base "r" number system (r>1)

- **The r's complement; "radix complement"**
- **The (r-1)'s complement; "diminished radix complement"**
- **Given a positive number N in base r, with integer part of n digits, and a fractional part of m digits**

$$N = d_{n-1}d_{n-2} \ldots d_2d_1d_0.d_{-1}d_{-2} \ldots d_{-m}$$

Complements

The r's complement is defined as N^*

$$\mathbf{N=d_{n-1}d_{n-2}\ ....d_2d_1d_0.d_{-1}d_{-2}\ ...d_{-m}}$$

$$\overset{*}{N} = 0 \; if \;\; N = 0$$

$$\overset{*}{N} = r^n - N \; if \;\; N > 0$$

Complements

The (r-1)' s complement is defined as $\bar{N}$

$N = d_{n-1} d_{n-2} \ldots. d_2 d_1 d_0 . d_{-1} d_{-2} \ldots d_{-m}$

$$\bar{N} = (r^n - r^{-m}) - N$$

NOTE: $$\overset{*}{N} = \bar{N} + r^{-m}$$

NOTE: if m=0 then $$\bar{N} = (r^n - 1) - N$$

$$\overset{*}{N} = \bar{N} + 1$$

Complements

Example: base 10 ($r = 10$)

- $N = 43.375$; $n = 2$; $m = 3$
- 10's complement
  - **$N^* = 10^2 - 43.375 = 100 - 43.375 = 56.625$**
- 9's complement
  - **$N^- = (10^2 - 10^{-3}) - 43.375 = 99.999 - 43.375 = 56.624$**
- Note: $N^* = N^- + r^{-3} = 56.624 + 0.001 = 56.625$

Complements

Example: base 2 ($r = 2$)

- $N = 101011.011$; $n = 6$; $m = 3$
- 2's complement
  - **$N^* = 2^6 - N = 1000000 - 101011.011 = 10100.101$**
- 1's complement
  - **$N^- = (2^6 - 2^{-3}) - N = 10100.1$**
- Note: $N^* = N^- + r^{-3} = 10100.1 + 0.001 = 10100.101$

Complements

Most of the time we work with integer numbers; i.e. no fractional part

- 2's complement

- Note: given a negative integer represented by its 2's complement, N*, we can obtain the magnitude of the integer by taking the 2's complement of N*

$$\overset{*}{N} = 0 \; if \; N = 0; \quad \overset{*}{N} = r^n - N \; if \; N > 0$$

$$or \quad N = r^n - \overset{*}{N}$$

Complements

Most of the time we work with integer numbers; i.e. no fractional part

- 1's complement

$$\bar{N} = (r^n - 1) - N$$

$$or \quad N = (r^n - 1) - \bar{N}$$

- Note: given a negative integer represented by its 1's complement, N^-, we can obtain the magnitude of the integer by taking the 1's complement of N^-

- NOTE: $N^* = \bar{N} + 1$

Complements: Algorithms

2' s complement

- **Leave all least significant zeros unchanged as well as the least significant one**
- **Replace 1' s by 0' s and 0' s by 1' s in all higher significant digits**

1' s complement

- **Flip every bit, replace 0' s by 1' s and 1' s by 0' s**

NOTE: 2' s complement = 1' s complement + 1

Subtraction using r’s complement

- Assume M and N are two positive numbers
- Want to perform the following operation M - N
- Algorithm
  - **Add M to the r’s complement of N**
    - **M – N = M + N***
  - **Inspect result for end carry**
    - **If end carry occurs, discard it**
    - **If no end carry occurs, then take the r’s complement of result and place a negative sign in front**

EXAMPLE: binary subtraction, 2's complement; 4-bit arithmetic

- 0110 M = 6_{10}
- -0011 N = -3_{10}
- Result is 3_{10}
- 0110 M
- + 1101 N*
- --------------
- 10011 carry not used, 0011 is correct answer

EXAMPLE: binary subtraction, 2's complement; 4-bit arithmetic

- 0011 M = 3_{10}
- -0111 N = -7_{10}
- Result is -4_{10}
- 0011 M
- + 1001 N*
- --------------
- 1100 no carry; take complement and add negative sign -0100

Subtraction using (r-1)' s complement

- Assume M and N are two positive numbers
- Want to perform the following operation M - N
- Algorithm
  - **Add M to the (r-1)' s complement of N**
    - **$M – N = M + N^-$**
  - **Inspect result for end carry**
    - **If end carry occurs, add 1 to LSB (end-around carry)**
    - **If no end carry occurs, then take the (r-1)' s complement of result and place a negative sign in front**

EXAMPLE: binary subtraction, 1's complement; 4-bit arithmetic

- 1001 M = 9_{10}
- -0100 N = -4_{10}
- Result is 5_{10}
- 1001 M
- + 1011 N^{-}
- --------------
- 10100 add 1 to 0100, result is 0101

EXAMPLE: binary subtraction, 1's complement; 4-bit arithmetic

- 0100 M = 4_{10}
- -1001 N = -9_{10}
- Result is -5_{10}
- 0100 M
- + 0110 N^{-}
- --------------
- 1010, no carry, take 1's complement 0101 and add negative sign in front, result is -0101

EXAMPLE: binary subtraction, 1's and 2's complement

Perform 1100 - 1100

- **1's complement**
  - **1100 + 0011 = (carry = 0) 1111 (positive zero)**
  - **Result = -0000 (negative zero)**
- **2's complement**
  - **1100 + 0100 = (carry = 1) 0000**
  - **Result = 0000**

OVERFLOW

When the word length is n bits, we say that an **overflow** has occurred if the correct representation of the sum (including sign) requires more than n bits (n+1 bits).

An overflow occurs if adding two positive numbers gives a negative answer or if adding two negative numbers gives a positive answer.

RANGE OF NUMBERS

NOTE: for a word length N

- The range of 2's complement numbers that can be represented is
    - -2^{N-1} to $(2^{N-1} - 1)$
- The range of 1's complement numbers that can be represented is
    - $-(2^{N-1} - 1)$ to $(2^{N-1} - 1)$

2's Complement Addition (a)
4-bit arithmetic

Range of numbers: -8 to +7

1. Addition of two positive numbers, sum $< 2^{n-1}$

$$\begin{array}{rrl} +3 & 0011 & \\ +4 & 0100 & \\ \hline +7 & 0111 & \text{(correct answer)} \end{array}$$

2. Addition of two positive numbers, sum $\geq 2^{n-1}$

$$\begin{array}{rrl} +5 & 0101 & \\ +6 & 0110 & \\ \hline & 1011 & \longleftarrow \text{wrong answer because of overflow (+11 requires} \\ & & \quad\text{5 bits including sign)} \end{array}$$

2's Complement Addition (b)
4-bit arithmetic

3. Addition of positive and negative numbers (negative number has greater magnitude)

+5	0101	
−6	1010	
−1	1111	(correct answer)

4. Same as case 3 except positive number has greater magnitude

−5	1011	
+6	0110	
+1	(1)0001	⟵ correct answer when the carry from the sign bit is ignored (this is *not* an overflow)

2's Complement Addition (c)
4-bit arithmetic

5. Addition of two negative numbers, $|\text{sum}| \le 2^{n-1}$

$$\begin{array}{rr} -3 & 1101 \\ -4 & 1100 \\ \hline -7 & (1)1001 \end{array} \quad \longleftarrow \text{ correct answer when the last carry is ignored}$$

(this is *not* an overflow)

6. Addition of two negative numbers, $|\text{sum}| > 2^{n-1}$

$$\begin{array}{rr} -5 & 1011 \\ -6 & 1010 \\ \hline & (1)0101 \end{array} \quad \longleftarrow \text{ wrong answer because of overflow}$$

(-11 requires 5 bits including sign)

1's Complement Addition (b)
4-bit arithmetic

3. Addition of positive and negative numbers (negative number with greater magnitude)

Range of numbers: -7 to +7
But have +0 and a -0

$+5$	0101	
-6	1001	
-1	1110	(correct answer)

4. Same as case 3 except positive number has greater magnitude

-5	1010	
$+6$	0110	
	(1) 0000	
	→ 1	(end-around carry)
	0001	(correct answer, *no* overflow)

1's Complement Addition (c)
4-bit arithmetic

5. Addition of two negative numbers, $|\text{sum}| < 2^{n-1}$

$$
\begin{array}{rrl}
-3 & 1100 & \\
\underline{-4} & \underline{1011} & \\
 & (1)\ 0111 & \\
 & \hookrightarrow 1 & \text{(end-around carry)} \\
 & \overline{1000} & \text{(correct answer, } \textit{no} \text{ overflow)}
\end{array}
$$

6. Addition of two negative numbers, $|\text{sum}| \geq 2^{n-1}$

$$
\begin{array}{rrl}
-5 & 1010 & \\
\underline{-6} & \underline{1001} & \\
 & (1)\ 0011 & \\
 & \hookrightarrow 1 & \text{(end-around carry)} \\
 & \overline{0100} & \text{(wrong answer because of overflow)}
\end{array}
$$

1's Complement Addition (d)
8-bit arithmetic

1. Add −11 and −20 in 1's complement.

$+11 = 00001011 \qquad\qquad +20 = 00010100$

taking the bit-by-bit complement,

−11 is represented by 11110100 and −20 by 11101011

$$\begin{array}{rr}
11110100 & (-11) \\
\underline{11101011} & \underline{+(-20)} \\
(1)\ 11011111 & \\
\xrightarrow{\qquad} 1 & \text{(end-around carry)} \\
\hline
11100000 = -31 &
\end{array}$$

2's Complement Addition (d)
8-bit arithmetic

2. Add -8 and $+19$ in 2's complement

$+8 = 00001000$

complementing all bits to the left of the first 1,
-8, is represented by 11111000

$$
\begin{array}{rr}
11111000 & (-8) \\
\underline{00010011} & \underline{+19} \\
\cancel{(1)}00001011 = & +11
\end{array}
$$

(discard last carry)

Signed Binary Numbers

Signed Binary Numbers

- Sign of a number must be represented with a 0 or a 1 in the leftmost position (by convention)
  - **0 = positive and 1 = negative**
- Digital system (Computer) does not care whether number is + or -, user is responsible for keeping track
- Not convenient for computers
- Number = (sign) (magnitude)
  - **n-bits = (1-bit) (n-1 bits)**
- Correction must be made when subtraction is performed

EXAMPLE: Signed-Magnitude System $(127)_{10}$

Use 8-bit arithmetic, limited by hardware (register)

Integer part:

127/2 = 63 + 1 (least significant bit - lsb)
63/2 = 31 + 1
31/2 = 15 + 1
15/2 = 7 + 1
7/2 = 3 + 1
3/2 = 1 + 1
1/2 = 0 + 1 (most significant bit - msb)

Read back in this direction

$(127)_{10}$ is then: sign = 0 magnitude = 1111111
$(-127)_{10}$ is then: sign = 1 magnitude = 1111111

EXAMPLE: Signed 1's complement $(127)_{10}$

Use 8-bit arithmetic, limited by hardware (register)

- **Positive numbers start with a 0, msb**
- **Negative numbers start with a 1, msb**
- **Convenient for digital systems**

$(127)_{10}$ is then: 01111111
$(-127)_{10}$ is then: 10000000
$(1)_{10}$ is then: 00000001
$(-1)_{10}$ is then: 11111110

$(11111111)_2$ is then $(-0)_{10}$
$(0000000)_2$ is then $(+0)_{10}$

NOTE: 1's complement has two representations of zero

EXAMPLE: Signed 2's complement $(127)_{10}$

Use 8-bit arithmetic, limited by hardware (register)

- **Positive numbers start with a 0, msb**
- **Negative numbers start with a 1, msb**
- **Convenient for digital systems**

$(127)_{10}$ is then: 01111111
$(-127)_{10}$ is then: 10000001
$(1)_{10}$ is then: 00000001
$(-1)_{10}$ is then: 11111111

$(11111111)_2$ is then $(-1)_{10}$
$(0000000)_2$ is then $(0)_{10}$

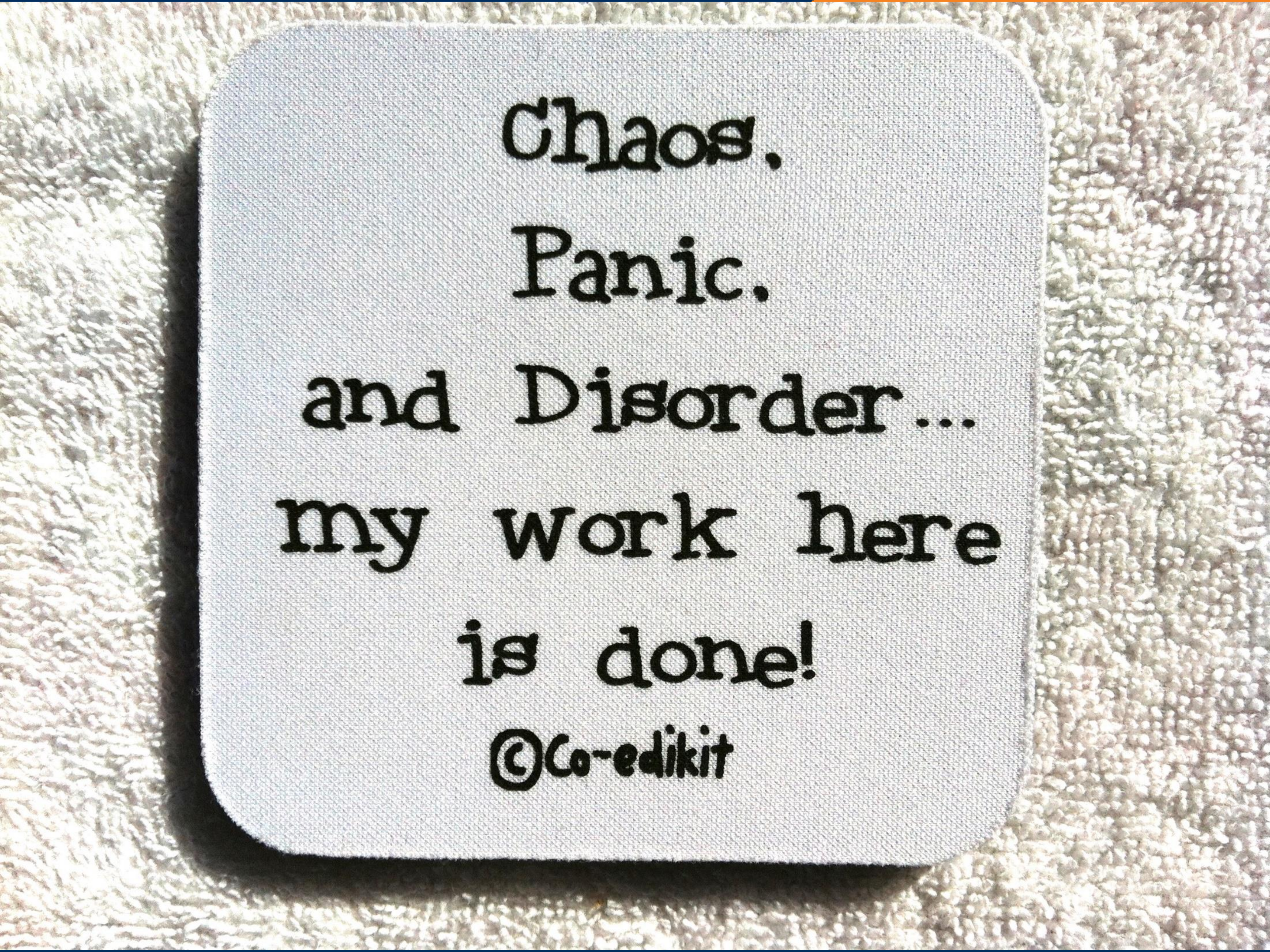
NOTE: 2's complement has one representation of zero

Signed Binary Integers (word length $n = 4$)

			Negative Integers		
$+N$	Positive Integers (all systems)	$-N$	Sign and Magnitude	2's Complement N^*	1's Complement $\bar{N}$
+0	0000	−0	1000	——	1111
+1	0001	−1	1001	1111	1110
+2	0010	−2	1010	1110	1101
+3	0011	−3	1011	1101	1100
+4	0100	−4	1100	1100	1011
+5	0101	−5	1101	1011	1010
+6	0110	−6	1110	1010	1001
+7	0111	−7	1111	1001	1000
		−8	——	1000	——

NOTE: for a word length N

- The range of 2's complement numbers that can be represented is
  - -2^{N-1} to $(2^{N-1} - 1)$
- The range of 1's complement numbers that can be represented is
  - $-(2^{N-1} - 1)$ to $(2^{N-1} - 1)$


Chaos.
Panic.
and Disorder...
my work here
is done!
©Co-edikit

Q&A