CALIFORNIA STATE UNIVERSITY
FULLERTON™

EGCP 281: Designing with VHDL

Fall 2024

Lecture 2: Binary Number and Arithmetic

Rakesh Mahto, Ph.D.
***Office**: E 314, California State University, Fullerton*
***Office Hours**: Monday and Wednesday 2:00 - 3:30 pm*
Or by appointment
***Office Hour Zoom Meeting ID**: 894 4126 5483*
***Email**: ramahto@fullerton.edu*
***Phone No**: 657-278-7274*

Binary Number Conversions

Binary Number Conversions

- The primary number systems used in digital systems are decimal (base-10), binary (base-2), octal (base-8) and hexadecimal (base-16)
- Octal and Hexadecimal number systems are used to represent binary numbers in a more compact form

Binary Codes for Decimal Digits

Decimal Digit	8-4-2-1 Code (BCD)	6-3-1-1 Code	Excess-3 Code	2-out-of-5 Code	Gray Code
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

Review of Decimal Number System

- Consider the symbol 9,437.25
  1. **9 – 1000s 9000** **most significant digit (MSD)**
  2. **4 – 100s 400**
  3. **3 – 10s 30**
  4. **7 – 1s 7**
  5. **2 – 1/10s .2**
  6. **5 – 1/100s .05** **least significant digit (LSD)**

Weighted-Positional Number System

- Add them up and you get 9,437.25
- To be more exact 9,437.25 should be written as: $9x10^3 + 4x10^2 + 3x10^1 + 7x10^0 + 2x10^{-1} + 5x10^{-2}$
- By convention we write only the coefficients $(9,437.25)_{10}$
  - **10 indicates the base, in this case is base 10 or decimal**
  - **Position of coefficient is a power (weight) of the base and specifies the information to be conveyed**

Review of Decimal Number System

- Consists of 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Coefficients are multiplied by powers of 10
- Said to be of base 10 or radix 10

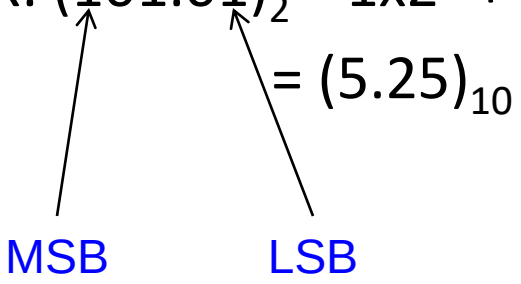
We can generalize to any base or radix "r".

- "r" is any integer greater than 1 (r > 1)
- Why r > 1?
- Consists of "r" possible values: 0, 1 , ….. r-1
- Coefficients are multiplied by powers of "r"

Binary Number System

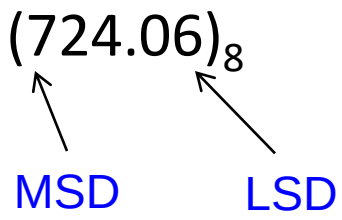
- Consists of two possible values (coefficients): 0 and 1
- Binary digits are called bits (short for binary digits)
- Byte = 8-bit binary number
- Nibble = 4-bit binary number
- Coefficients are multiplied by powers of 2
- Binary system is of base 2 or radix 2
- EX: $(101.01)_2 = 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2}$

  $= (5.25)_{10}$



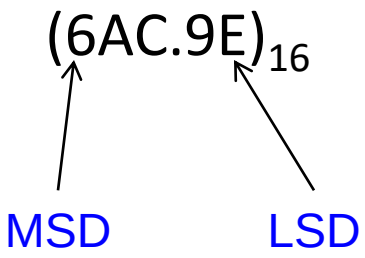
Octal Number System

- Consists of eight possible values (coefficients): 0, 1, 2, 3, 4, 5, 6, and 7
- Coefficients are multiplied by powers of 8
- Octal system is of base 8 or radix 8
- EX: $(724.06)_8 = 7x8^2 + 2x8^1 + 4x8^0 + 0x8^{-1} + 6x8^{-2} = 448 + 16 + 4 + 0.015625 = (468.015625)_{10}$



Hexadecimal Number System

- Consists of sixteen possible values (coefficients): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Coefficients are multiplied by powers of 16
- Hexadecimal system is of base 16 or radix 16
- EX: $(6AC.9E)_{16} = 6x16^2 + Ax16^1 + Cx16^0 + 9x16^{-1} + Ex16^{-2} = 1536 + 160 + 12 + 0.5 + 0.05468 = (1708.55468)_{10}$



Binary, Octal and Hexadecimal

NOTE: binary, octal and hex numbers are all powers of 2

- $2^3 = 8$: each octal digit corresponds to 3 binary digits; with 3 binary digits we can represent 8 combinations; 000 to 111, 0 to 7 decimal
- $2^4 = 16$: each hexadecimal digit corresponds to 4 binary digits; with 4 binary digits we can represent 16 combinations; 0000 to 1111, 0 to F (15) decimal

Octal and Hex are easier to read and more compact to display than Binary

Number Base Conversions

Number of Base "r" to Decimal Number

$(number)_r \longrightarrow (number)_{10}$

Any base to decimal: form products by multiplying the coefficients times appropriate powers of the base, and then sum the products

EX: $(101.01)_2 = 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} = (5.25)_{10}$

EX: $(724.06)_8 = 7x8^2 + 2x8^1 + 4x8^0 + 0x8^{-1} + 6x8^{-2} = 448 + 16 + 4 + 0.015625 = (468.015625)_{10}$

EX: $(6AC.9E)_{16} = 6x16^2 + Ax16^1 + Cx16^0 + 9x16^{-1} + Ex16^{-2} = 1536 + 160 + 12 + 0.5 + 0.05468 = (1708.55468)_{10}$

Decimal Number to Number of Base "r"

$(\text{number})_{10} \longrightarrow (\text{number})_r$

We will use an Algorithm: efficient procedure (sequence of steps), recipe

Decimal to any base:

1. Separate number into an integer part and a fraction part
2. Integer portion is repeatedly divided by the base to which we are converting. NOTE: remainders are the appropriate coefficients
3. Fractional part is repeatedly multiplied by the base to which we are converting. NOTE: integer portions of resulting numbers are the appropriate coefficients
4. Combine the resulting answers from 2 and 3

EXAMPLE: Convert $(25.375)_{10}$ to base 2 (binary)

Integer = 25 **Fraction = 0.375**

Integer part:

25/2 = 12 + 1 (least significant bit - lsb)
12/2 = 6 + 0
6/2 = 3 + 0
3/2 = 1 + 1
1/2 = 0 + 1 (most significant bit - msb)

Read back in this direction

$(25)_{10} = (11001)_2$

EXAMPLE: Convert $(25.375)_{10}$ to base 2 (binary)

Integer = 25 **Fraction = 0.375**

Fraction part:

.375 * 2 = .75 + 0 (most significant bit - msb)
.75 * 2 = .5 + 1
.5 * 2 = 0 + 1 (least significant bit - lsb)

Read back in this direction

$(0.375)_{10} = (011)_2$

Result: $(25.375)_{10} = (11001.011)_2$

EXAMPLE: Convert 53_{10} to binary.

$$
\begin{array}{rll}
2 \,\big/\!\underline{53} & & \\
2 \,\big/\!\underline{26} & \text{rem.} = 1 = a_0 & \\
2 \,\big/\!\underline{13} & \text{rem.} = 0 = a_1 & \\
2 \,\big/\!\underline{6} & \text{rem.} = 1 = a_2 & 53_{10} = 110101_2 \\
2 \,\big/\!\underline{3} & \text{rem.} = 0 = a_3 & \\
2 \,\big/\!\underline{1} & \text{rem.} = 1 = a_4 & \\
0 & \text{rem.} = 1 = a_5 &
\end{array}
$$

Conversion (a)

EXAMPLE: Convert $.625_{10}$ to binary.

$$F = .625 \times 2 = 1.250 \quad (a_{-1} = 1)$$

$$F_1 = .250 \times 2 = 0.500 \quad (a_{-2} = 0)$$

$$F_2 = .500 \times 2 = 1.000 \quad (a_{-3} = 1)$$

$$.625_{10} = .101_2$$

Conversion (b)

EXAMPLE: Convert 0.7_{10} to binary.

.7	
2	
(1).4	
2	
(0).8	
2	
(1).6	
2	
(1).2	
2	
(0).4	←— process starts repeating here because 0.4 was previously obtained
2	
(0).8	$0.7_{10} = 0.1\ \underline{0110}\ \underline{0110}\ \underline{0110}\ \ldots_2$

Conversion (c)

EXAMPLE: Convert $(720.75)_{10}$ to base 8 (octal)

Integer = 720 Fraction = 0.75

Integer part:

720/8 = 90 + 0 (least significant bit - lsd)
90/8 = 11 + 2
11/8 = 1 + 3
1/8 = 0 + 1 (most significant bit - msd)

Read back in this direction

$(720)_{10} = (1320)_8$

EXAMPLE: Convert $(720.75)_{10}$ to base 8 (octal)

Integer = 720 Fraction = 0.75

Fraction part:

.75 * 8 = .0 + 6 (most significant bit - msd)
and (least significant bit - lsd)

Read back in this direction

$(0.75)_{10} = (0.6)_8$

Result: $(720.75)_{10} = (1320.6)_8$

EXAMPLE: Convert 231.3_4 to base 7.

$$231.3_4 = 2 \times 16 + 3 \times 4 + 1 + \tfrac{3}{4} = 45.75_{10}$$

$7 \overline{)45}$		.75
$7 \overline{)\ 6}$	rem. 3	7
0	rem. 6	(5) .25
		7
		(1) .75
		7
		(5) .25
		7
		(1) .75

$$45.75_{10} = 63.5151\ldots_7$$

Conversion (d)

Binary, Octal and Hexadecimal Conversions

How to Convert Between Binary, Octal and Hexadecimal

Conversion of base is done by partitioning

- If needed add zeros to extreme left when converting the integer portion
- Add zeros to extreme right when converting fractional part
- Binary to Octal
  - **Partition binary number into groups of three digits starting from binary point**
- Octal to Binary
  - **Each octal digit is converted to its 3-digit binary equivalent**
- Binary to Hexadecimal
  - **Partition binary number into groups of four digits starting from binary point**
- Hexadecimal to Binary
  - **Each hex digit is converted to its 4-digit binary equivalent**

How to Convert Between Binary, Octal and Hexadecimal

Binary to Octal

- **Partition binary number into groups of three digits starting from binary point**
- **$(101100.01101)_2$**
- **101 100. 011 010**
- **5 4. 3 2**
- **$(54.32)_8$**

Octal to Binary

- **Each octal digit is converted to its 3-digit binary equivalent**
- **$(3.1)_8$**
- **3. 1**
- **011. 001**
- **$(011.001)_2$**

How to Convert Between Binary, Octal and Hexadecimal

Binary to Hexadecimal

- **Partition binary number into groups of four digits starting from binary point**
- **$(111010.011)_2$**
- **0011 1010. 0110**
- **3 A. 6**
- **$(3A.6)_{16}$**

Hexadecimal to Binary

- **Each hex digit is converted to its 4-digit binary equivalent**
- **$(6.D)_{16}$**
- **6. D**
- **0110. 1101**
- **$(0110.1101)_2$**

Binary ⇔ Hexadecimal Conversion

$$1001101.010111_2 = \underbrace{0100}_{4} \quad \underbrace{1101}_{D} . \underbrace{0101}_{5} \quad \underbrace{1100}_{C} = 4D.5C_{16}$$

Conversion from binary to hexadecimal (and conversely) can be done by inspection because each hexadecimal digit corresponds to exactly four binary digits (bits).

How about conversion between Octal and Hexadecimal and vice versa?

EXAMPLE: Convert $(31.7)_8$ to Hex.

Octal to Hexadecimal

- $(31.7)_8$
- 31. 7
- 011 001. 111 (binary equivalent)
- 0001 1001. 1110 (group digits by four)
- $(19.E)_{16}$

EXAMPLE: Convert $(AB.6)_{16}$ to octal.

Hexadecimal to Octal

- $(AB.6)_{16}$
- A B. 6
- 1010 1011. 0110 (binary equivalent)
- 010 101 011. 011 (group digits by three)
- $(253.3)_8$

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, …, 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:

(a)
```
   1<---- (indicates
  11101   a borrow
 -10011   from the
  ------
   1010   3rd column)
```

(b)
```
  1111<---- borrows
  10000
-    11
  -----
  1101
```

(c)
```
   111<---- borrows
  111001
-   1011
  ------
  101110
```

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:

$$
\begin{aligned}
205 - 18 &= [2 \times 10^2 + 0 \times 10^1 + 5 \times 10^0] \\
&\quad - [\qquad 1 \times 10^1 + 8 \times 10^0] \\
&\qquad\qquad \text{note borrow from column 1} \\
&= [2 \times 10^2 + (0 - 1) \times 10^1 + (10 + 5) \times 10^0] \\
&\quad - [\qquad 1 \times 10^1 + \qquad 8 \times 10^0] \\
&\qquad\qquad \text{note borrow from column 2} \\
&= [(2 - 1) \times 10^2 + (10 + 0 - 1) \times 10^1 + 15 \times 10^0] \\
&\quad - [\qquad\qquad 1 \times 10^1 + 8 \times 10^0] \\
&= [1 \times 10^2 \quad + \quad 8 \times 10^1 \quad + \quad 7 \times 10^0] = 187
\end{aligned}
$$

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:

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

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^*

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

$$N^* = 0 \; if \; N = 0$$

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

Complements

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

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

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

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

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

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

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

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

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

+3	0011	
+4	0100	
+7	0111	(correct answer)

Range of numbers: -8 to +7

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

+5	0101	
+6	0110	
	1011	⟵ wrong answer because of overflow (+11 requires 5 bits including sign)

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, $|\text{sum}| \leq 2^{n-1}$

-3	1101	
-4	1100	
-7	(1)1001	⟵ correct answer when the last carry is ignored (this is *not* an overflow)

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

-5	1011	
-6	1010	
	(1)0101	⟵ 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, $|\text{sum}| < 2^{n-1}$

-3	1100	
-4	1011	
(1)	0111	
$\hookrightarrow$	1	(end-around carry)
	1000	(correct answer, *no* overflow)

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

-5	1010	
-6	1001	
(1)	0011	
$\hookrightarrow$	1	(end-around carry)
	0100	(wrong answer because of overflow)

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

1. Add −11 and −20 in 1's complement.

+11 = 00001011 +20 = 00010100

taking the bit-by-bit complement,

−11 is represented by 11110100 and −20 by 11101011

```
  11110100     (−11)
  11101011   +(−20)
 ---------
(1) 11011111
  └──────→ 1      (end-around carry)
 ---------
  11100000 = −31
```

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

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

↑ (discard last carry)

Signed Binary Numbers

Signed Binary Numbers

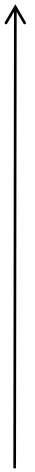
- Sign of a number must be represented with a 0 or a 1 in the leftmost position (by convention)
  - **0 = positive and 1 = negative**
- Digital system (Computer) does not care whether number is + or -, user is responsible for keeping track
- Not convenient for computers
- Number = (sign) (magnitude)
  - **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 $(0)_{10}$

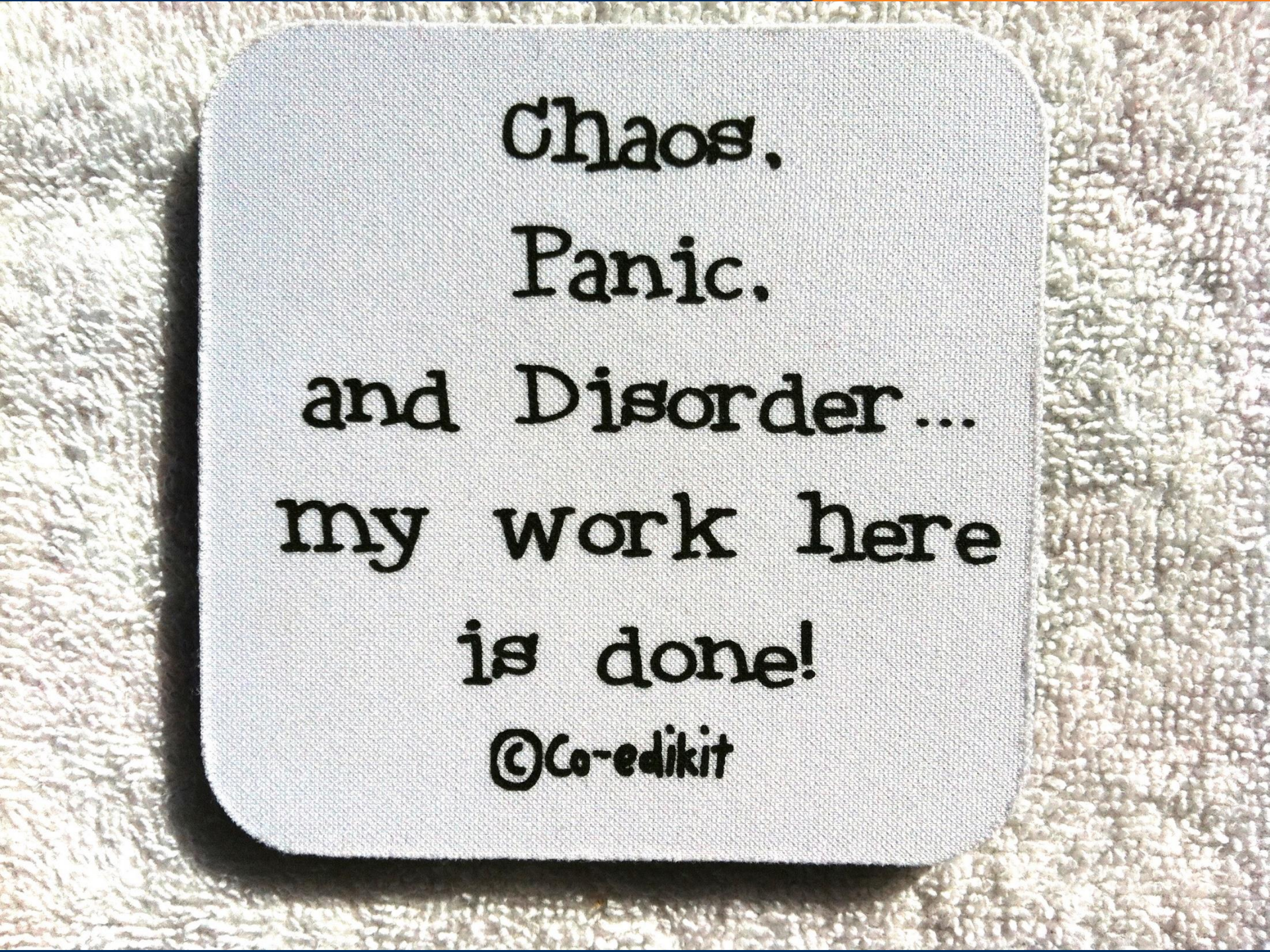
NOTE: 2's complement has one representation of zero

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

$+N$	Positive Integers (all systems)	$-N$	Negative Integers: Sign and Magnitude	Negative Integers: 2's Complement N^*	Negative Integers: 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