Tutorial 02: Signed Numbers

Computer Science Department
CS2208: Introduction to Computer Organization and Architecture
Winter 2020-2021
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Signed Numbers

- Computer designers have adopted various techniques to represent negative numbers, including
  - ***sign and magnitude***,
  - ***biased representation***, and
  - ***two's complement***.

Sign and Magnitude

- *Example 1*: Convert -743_8 to binary using ***sign and magnitude*** method

743_8

$\rightarrow 111\ \ 100\ \ 011_2$

$\rightarrow 111100011_2$

unsigned value

0 = 000
1 = 001
2 = 010
3 = 011
4 = 100
5 = 101
6 = 110
7 = 111

-743_8

$\rightarrow 1111100011_2$

Sign and Magnitude

- *Example 2*: Convert $-AB.BA_{16}$ to binary using ***sign and magnitude*** method

$AB.BA_{16}$

$\rightarrow 1010\ \ 1011.1011\ \ 1010_2$

$\rightarrow 10101011.1011101_2$

unsigned value

$-AB.BA_{16}$

$\rightarrow 110101011.1011101_2$

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
A = 1010
B = 1011
C = 1100
D = 1101
E = 1110
F = 1111

Sign and Magnitude

- *Example 3*: Convert $-0.0A_{16}$ to binary using ***sign and magnitude*** method

$0.0A_{16}$

$\rightarrow 0000.0000\ 1010_2$

$\rightarrow 0.0000101_2$

unsigned value

$-0.0A_{16}$

$\rightarrow 10.0000101_2$

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
A = 1010
B = 1011
C = 1100
D = 1101
E = 1110
F = 1111

Biased Representation

- ***Example 4***: Encode -14_{10} using ***excess-32*** representation method (a.k.a. ***biased representation***)

To encode a number using ***excess-32*** method, you need to add 32 to that number.

- $-14_{10} + 32_{10} = 18_{10}$
- 18_{10} is the ***excess-32*** representation of -14_{10}

To decode an ***excess-32*** value to its original value, you need to subtract 32.

- $18_{10} - 32_{10} = -14_{10}$

Biased Representation

- ***Example 5***: Encode 14_{10} using ***excess-127*** representation method (a.k.a. ***biased representation***)

To encode a number using ***excess-127*** method, you need to add 127 to that number.

- $14_{10} + 127_{10} = 141_{10}$
- 141_{10} is the ***excess-127*** representation of 14_{10}

To decode an ***excess-127*** value to its original value, you need to subtract 127.

- $141_{10} - 127_{10} = 14_{10}$

2's Complement

- In binary arithmetic, the *two's complement* of a number is formed by
  - *Subtracting the number from 2^n.*

    The *two's* complement of 01100101_2 is
    $100000000_2 - 01100101_2 = 10011011_2$

  - *Flipping (inverting) all the bits of the number* and *adding 1*.

    The *two's* complement of 01100101_2 is
    $10011010_2 + 1_2 = 10011011_2$.

  - *Processing all the bits of the number from the least significant bit (LSB) towards the most significant bit (MSB)*
    - copying all the zeros until the first 1 is reached,
    - copying that 1,
    - flipping (inverting) all the remaining bits.

    The *two's* complement of 01100100_2 is 10011100_2.
    The *two's* complement of 01100101_2 is 10011011_2.

In binary system, the sign is encoded as:
MSD = 0 → positive
MSD = 1 → negative

Just for the sake of completeness, in radix R systems, the sign is encoded as:
MSD < R/2 → positive
MSD ≥ R/2 → negative.

2's Complement

- ***Example 6***: Convert $-AB.BA_{16}$ to binary using ***2's complement*** method

$AB.BA_{16}$

$\rightarrow 1010\ \ 1011.1011\ \ 1010_2$

$\rightarrow 10101011.1011101_2$

unsigned value

$+AB.BA_{16}$

$\rightarrow 010101011.1011101_2$

$-AB.BA_{16}$

$\rightarrow 101010100.0100011_2$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

2's Complement

- ***Example 7***: Convert $-0.0A_{16}$ to binary using ***2's complement*** method

$0.0A_{16}$

$\rightarrow 0000.0000\ \ 1010_2$

$\rightarrow 0.0000101_2$

unsigned value

$+0.0A_{16}$

$\rightarrow 00.0000101_2$

$-0.0A_{16}$

$\rightarrow 11.1111011_2$

0 = 0000
1 = 0001
2 = 0010
3 = 0011
4 = 0100
5 = 0101
6 = 0110
7 = 0111
8 = 1000
9 = 1001
A = 1010
B = 1011
C = 1100
D = 1101
E = 1110
F = 1111

Signed Numbers

Binary pattern	Unsigned	Signed-and-magnitude	2's complement	Excess-8
0000	0	+0	+0	– 8
0001	1	+1	+1	– 7
0010	2	+2	+2	– 6
0011	3	+3	+3	– 5
0100	4	+4	+4	– 4
0101	5	+5	+5	– 3
0110	6	+6	+6	– 2
0111	7	+7	+7	– 1
1000	8	– 0	– 8	+ 0
1001	9	– 1	– 7	+1
1010	10	– 2	– 6	+2
1011	11	– 3	– 5	+3
1100	12	– 4	– 4	+4
1101	13	– 5	– 3	+5
1110	14	– 6	– 2	+6
1111	15	– 7	– 1	+7

For a given *n* bit binary pattern

*What is the **number of zeros** for various values of n?*

*What is the **range** for various values of n?*

	Unsigned	Signed-and-magnitude	2's complement	Excess-8
Number of zeros	1	2	1	1
Range	$0 \rightarrow 2^n - 1$	$-(2^{n-1} - 1) \rightarrow 2^{n-1} - 1$	$-(2^{n-1}) \rightarrow 2^{n-1} - 1$	$-(2^{n-1}) \rightarrow 2^{n-1} - 1$

Unsigned

- ***Example 8***: Convert 11011.11011_2 to decimal, *assuming* that it is an ***unsigned*** number.

$11011_2 \rightarrow 27_{10}$

$0.11011_2 \rightarrow 0.84375_{10}$

$11011.11011_2 \rightarrow 27.84375_{10}$

Another method:

$$11011.11011_2 = 1101111011_2 / 100000_2$$
$$= 891_{10} / 32_{10}$$
$$= 27.84375_{10}$$

Sign and Magnitude

- *Example 9*: Convert 11011.11011_2 to decimal, *assuming* that it is encoded using ***sign and magnitude*** method.

$11011.11011_2 \rightarrow -1011.11011_2$

$1011_2 \rightarrow 11_{10}$

$0.11011_2 \rightarrow 0.84375_{10}$

$1011.11011_2 \rightarrow 11.84375_{10}$

unsigned value

$11011.11011_2 \rightarrow -11.84375_{10}$

Another method:

$11011.11011_2 \rightarrow -1011.11011_2$

$1011.11011_2 = 101111011_2 / 100000_2$

$= 379_{10} / 32_{10} = 11.84375_{10}$

unsigned value

$11011.11011_2 \rightarrow -11.84375_{10}$

2's Complement

- *Example 10*: Convert 11011.11011_2 to decimal, *assuming* that it is encoded using ***2's complement*** method.

$11011.11011_2 \rightarrow$ *negative number*

$11011.11011_2 \rightarrow -00100.00101_2$

$00100_2 \rightarrow 4_{10}$

$0.00101_2 \rightarrow 0.15625_{10}$

$00100.00101_2 \rightarrow 4.15625_{10}$

$11011.11011_2 \rightarrow -4.15625_{10}$

unsigned value

Another method:

$11011.11011_2 \rightarrow$ *negative number*

$11011.11011_2 \rightarrow -00100.00101_2$

$00100.00101_2 = 0010000101_2 / 100000_2$

$= 133_{10} / 32_{10} = 4.15625_{10}$

$11011.11011_2 \rightarrow -4.15625_{10}$

unsigned value

2's Complement

- The following numbers represent the same value, which is $+14_{10}$
  - 01110_2
  - 001110_2
  - 0001110_2
  - 00001110_2
  - 000001110_2
  - 0000001110_2
  - …

+14 using 5, 6 ,7, 8, 9, and 10 bits

Basically, the sign is extended

- By Converting these numbers into the *2's complement*, you get
  - 10010_2
  - 110010_2
  - 1110010_2
  - 11110010_2
  - 111110010_2
  - 1111110010_2
  - …

-14 in 2's complement using 5, 6 ,7, 8, 9, and 10 bits

Basically, the sign is extended

2's Complement

- ***Example 11***: Convert **11011_2** to decimal, *assuming* that it is encoded using ***2's complement*** method.

- 11011_2 ➔ *negative number*
- 11011_2 ➔ -00101_2
- 00101_2 ➔ 5_{10}
- 11011_2 ➔ -5_{10}

2's Complement

- *Example 12*: Convert **1111011_2** to decimal, *assuming* that it is encoded using ***2's complement*** method.

- 1111011_2 ➔ *negative number*
- 1111011_2 ➔ -0000101_2
- 0000101_2 ➔ 5_{10}
- 1111011_2 ➔ -5_{10}

2's Complement

- ***Example 13***: Convert **111111011_2** to decimal, *assuming* that it is encoded using ***2's complement*** method.

- 111111011_2 → *negative number*
- 111111011_2 → -000000101_2
- 000000101_2 → 5_{10}
- 111111011_2 → -5_{10}