Tutorial 01: Number Systems and Signed Numbers

Computer Science Department
CS2208b: Fundamentals of Computer Organization and Architecture
Winter 2018
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Number Systems

- A number system
  - Is a notation for ***representing*** (***encoding***) numbers in a consistent manner.
  - Can be based on
    - positional notation (i.e., place-value),
      - Using the same symbol for the different orders of magnitude
        - for example, in decimal, we have the *ones place*, *tens place*, *hundreds place*.
    - other notations (e.g., Roman numerals)

Number Systems

- A radix or base is
  - the number of unique digits, ***including zero***, used to represent numbers in a positional numeral system.
- With the use of a *radix point*, the notation can be extended to include *fractions* and *real numbers*.

Number Systems

- Examples of positional numeral systems
  - Decimal is base-10 → {0, 1, 2, 3, 4, 5, 6, 7, 8, and 9}

  - Binary is base-2 → {0, and 1}
  - Quaternary is base-4 → {0, 1, 2, and 3}
  - Octal is base-8 → {0, 1, 2, 3, 4, 5, 6, and 7}
  - Hexadecimal is base-16 → {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F}

  - Trinary is base-3 → {0, 1, and 2}
  - Quinary is base-5 → {0, 1, 2, 3, and 4}
  - Senary is base-6 → {0, 1, 2, 3, 4, and 5}
  - Septenary is base-7 → {0, 1, 2, 3, 4, 5, and 6}
  - Nonary is base-9 → {0, 1, 2, 3, 4, 5, 6, 7, and 8}
  - Duodecimal is base-12 → {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, and B}
  - Sexagesimal is base-60 → {0, 1, 2, 3, 4, 5, …, 58 and 59}

You need to know how to convert values from one system to the other

Conversion from any other base system to decimal

- If the original number in base *b* is:

$$(a_{n-1}\ a_{n-2}\ \ldots\ a_i\ \ldots\ a_1\ a_0\ .\ a_{-1}\ a_{-2}\ \ldots\ a_{-m})_b$$

- The *decimal* value of this number is defined as

$$N_{10} = (a_{n-1}b^{n-1} + \ldots + a_i b^i + \ldots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots + a_{-m} b^{-m})_{10}$$

Conversion from any other base system <u>to decimal</u>

- ***Example 1***: Convert $2E8_{16}$ to *decimal*

$$
\begin{aligned}
2E8_{16} &= 2\times16^2 + E\times16^1 + 8\times16^0 \\
&= 2\times256 + 14\times16 + 8\times1 \\
&= 512 + 224 + 8 \\
&= 744_{10}
\end{aligned}
$$

10 = A
11 = B
12 = C
13 = D
14 = E
15 = F

Conversion from any other base system to decimal

- ***Example 2***: Convert 361_8 to *decimal*

$$361_8 = 3\times8^2 + 6\times8^1 + 1\times8^0$$
$$= 3\times64 + 6\times8 + 1\times1$$
$$= 192 + 48 + 1$$
$$= 241_{10}$$

Conversion from any other base system <u>to decimal</u>

- ***<u>Example 3</u>***: Convert 0.361_8 to *decimal*

$0.361_8 = 3\times8^{-1} + 6\times8^{-2} + 1\times8^{-3}$

$= 3\times0.125 + 6\times0.015625 + 1\times0.001953125$

$= 0.375 + 0.09375 + 0.001953125$

$= 0.470703125_{10}$

<u>Another method:</u>

$0.361_8 = 361_8 / 1000_8$

$= (3\times8^2 + 6\times8^1 + 1\times8^0) / (1\times8^3)$

$= (3\times64 + 6\times8 + 1\times1) / (1\times512)$

$= (192 + 48 + 1) / (512)$

$= 241 / 512$

$= 0.470703125_{10}$

Conversion from any other base system to decimal

- ***Example 4***: 12.112_3 to *decimal*

12.112_3

$= 1\times3^{1} + 2\times3^{0} + 1\times3^{-1} + 1\times3^{-2} + 2\times3^{-3}$

$= 1 \times 3 + 2 \times 1 + 1 \times 0.33333 + 1 \times 0.11111 + 2 \times 0.03703$

$= 3 + 2 + 0.33333 + 0.11111 + 0.07406 = 5.5185_{10}$

Another method:

$12.112_3 = 12112_3 / 1000_3$

$= (1\times3^4 + 2\times3^3 + 1\times3^2 + 1\times3^1 + 2\times3^0) / (1\times3^3)$

$= (81 + 54 + 9 + 3 + 2) / 27$

$= 149 / 27 = 5.5185_{10}$

Conversion from decimal to any other base system

- ***Division Method (for integer numbers)***
  - Initialize the quotient by the value of the *decimal number*
  - *Repeat*:
    - Divide the quotient from the previous stage by the new base to get
      - A quotient (the whole number)
      - A *remainder*
    - The *remainder* here is *the next least significant digit* in the new number

    Until the quotient becomes 0.

Conversion from decimal to any other base system

- *Example 5*: Convert 14_{10} to binary

  Binary means the new base is 2

  - $14/2 = 7$ Remainder: 0 ➔ This is the least significant binary digit
    Quotient = $7 \neq 0$ ➔ continue
  - $7/2 = 3$ Remainder: 1 ➔ This is the 2nd least significant binary digit
    Quotient = $3 \neq 0$ ➔ continue
  - $3/2 = 1$ Remainder: 1 ➔ This is the 3rd least significant binary digit
    Quotient = $1 \neq 0$ ➔ continue
  - $1/2 = 0$ Remainder: 1 ➔ This is the 4th least significant binary digit
    Quotient = 0 ➔ exit the *repeat-until* control structure
  - $14_{10} = 1110_2$

Note that, it is 1110_2
It is NOT 0111_2

Conversion from decimal to any other base system

- ***Example 6***: Convert 2477_{10} to hexadecimal:

  Hexadecimal means the new base is 16

  - $2477/16 = 154$ Remainder: 13 ➔ This is the least significant Hex digit
    Quotient = $154 \neq 0$ ➔ continue
  - $154/16 = 9$ Remainder: 10 ➔ This is the 2nd least significant Hex digit
    Quotient = $9 \neq 0$ ➔ continue
  - $9/16 = 0$ Remainder: 9 ➔ This is the 3rd least significant Hex digit
    Quotient = 0 ➔ exit the *repeat-until* control structure
  - $2477_{10} = 9AD_{16}$

Note that, it is $9AD_{16}$
It is NOT $DA9_{16}$

10 = A	13 = D
11 = B	14 = E
12 = C	15 = F

Conversion from decimal to any other base system

- ***Multiplication Method (for fraction numbers)***
  - Initialize the fraction by the value of the *fractional decimal number*
  - *Repeat*:
    - Multiply the fraction from the previous stage by the new base to get
      - A *whole number*
      - A *fraction*
    - The *whole number* here is *the next digit to the right after the radix point* in the new number

  Until the fraction becomes 0.

This condition might be *reached* or *not*.

If reached,

then we will have 100% accurate result.

If not,

then we have to approximate the result.

Conversion <u>from decimal</u> to any other base system

- ***Example 7***: Convert 0.017578125_{10} to hexadecimal

  Hexadecimal means the new base is 16

  - $0.01757812 \times 16 = 0.28125$

    whole number: 0 ➔ *the next digit to the right after the radix point*

    fraction $= 0.28125 \neq 0$ ➔ continue

  - $0.28125 \times 16 = 4.5$

    whole number: 4 ➔ *the next digit to the right after the radix point*

    fraction $= 0.5 \neq 0$ ➔ continue

  - $0.5 \times 16 = 8.0$

    whole number: 8 ➔ *the next digit to the right after the radix point*

    fraction $= 0.0$ ➔ exit the *repeat-until* control structure

  - $0.017578125_{10} = 0.048_{16}$

Conversion from decimal to any other base system

- ***Example 8***: Convert 255.017578125_{10} to hexadecimal:

  Hexadecimal means the new base is 16

  $255.017578125_{10} = 255_{10} + 0.017578125_{10}$

  Using the ***division method***: $255_{10} \rightarrow FF_{16}$

  Using the ***multiplication method***: $0.017578125_{10} \rightarrow 0.048_{16}$

$255.017578125_{10} = FF.048_{16}$

Conversion from decimal to any other base system

- *Example 9*: Convert 0.85_{10} to hexadecimal
  Hexadecimal means the new base is 16
  - $0.85 \times 16 = 13.6$
    whole number: 13 ➔ *the next digit to the right after the radix point*
    fraction $= 0.6 \neq 0$ ➔ continue
  - $0.6 \times 16 = 9.6$
    whole number: 9 ➔ *the next digit to the right after the radix point*
    fraction $= 0.6 \neq 0$ ➔ continue
  - $0.6 \times 16 = 9.6$
    whole number: 9 ➔ *the next digit to the right after the radix point*
    fraction $= 0.6 \neq 0$ ➔ continue
  - …
  - $0.85_{10} = 0.D99999\ldots9_{16}$
  - Can be approximated in 4 digits after the radix point, for example, as
    - ***$0.D999_{16}$ (using truncation)*** or as
    - ***$0.D99A_{16}$ (using rounding)***

Conversion between any two bases, other than decimal

- This task can be done in two steps:
  - Convert from the source base to the decimal
  - Convert from the decimal to the destination base

Conversion between any two bases, other than decimal

10 = A	13 = D
11 = B	14 = E
12 = C	15 = F

- ***Example 10***: Convert $2E8_{16}$ to *octal*

$$2E8_{16} = 2\times16^2 + E\times16^1 + 8\times16^0$$
$$= 2\times256 + 14\times16 + 8\times1$$
$$= 512 + 224 + 8 = 744_{10}$$

744/8 = 93 Remainder: 0 ➔ This is the least significant octal digit
Quotient = 93 ≠ 0 ➔ continue

93/8 = 11 Remainder: 5 ➔ This is the 2nd least significant octal digit
Quotient = 11 ≠ 0 ➔ continue

11/8 = 1 Remainder: 3 ➔ This is the 3rd least significant octal digit
Quotient = 1 ≠ 0 ➔ continue

1/8 = 0 Remainder: 1 ➔ This is the 4th least significant octal digit
Quotient = 11 ≠ 0 ➔ exit the *repeat-until* control structure

$$\mathbf{2E8_{16} = 744_{10} = 1350_8}$$

Conversion between any two bases, other than decimal (Special cases)

- ***Binary to octal or hexadecimal:***
  - ***Binary to octal conversion***
    - Group bits in three's, *starting from the binary point* (*pad the last group with 0's, if needed*)
  - ***Binary to hexadecimal conversion***
    - Group bits in four's, *starting from the binary point* (*pad the last group with 0's, if needed*)

Conversion between <u>any two bases</u>, <u>other than decimal</u> (Special cases)

- ***Example 11***: Convert 11001111_2 to *octal*

11001111_2

→ $011\ \ 001\ \ 111_2$

→ 317_8

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

Conversion between any two bases, other than decimal (Special cases)

- ***Example 12***: Convert 1111010101_2 to hexadecimal

1111010101_2

$\rightarrow$ $0011\ 1101\ 0101_2$

$\rightarrow$ $3D5_{16}$

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

Conversion between any two bases, other than decimal (Special cases)

- ***Octal or hexadecimal to binary:***
  - ***Octal to binary conversion***
    - Expanding each octal digit into three bits
  - ***Hexadecimal to binary conversion***
    - Expanding each hexadecimal digit into four bits

Conversion between any two bases, other than decimal (Special cases)

- ***Example 13***: Convert 743_8 to binary

743_8

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

$\rightarrow 111100011_2$

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

Conversion between <u>any two bases</u>, <u>other than decimal</u> (Special cases)

- ***Example 14***: Convert $FA9_{16}$ to binary

$FA9_{16}$

$\rightarrow 1111 \quad 1010 \quad 1001_2$

$\rightarrow 111110101001_2$

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

Conversion between <u>any two bases</u>, <u>other than decimal</u> (Special cases)

- ***<u>Octal to hexadecimal</u> or <u>hexadecimal to octal:</u>***
  - Convert from the source base to the binary
    - Expanding each digit into three bits (in case of octal) or four bits (in case of octal)
  - Convert from the binary to the destination base
    - Group bits in three's (in case of octal) or four's (in case of hexadecimal), *<u>starting from the binary point</u>* (*pad the last group <u>from both sides</u> with 0's, if needed*)

Conversion between any two bases, other than decimal (Special cases)

- ***Example 15***: Convert ABC_{16} to octal

ABC_{16}

$\rightarrow$ $1010\ 1011\ 1100_2$

$\rightarrow$ 101010111100_2

$\rightarrow$ $101\ 010\ 111\ 100_2$

$\rightarrow$ 5274_8

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

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

Conversion between any two bases, other than decimal (Special cases)

- ***Example 16***: Convert $0.AB_{16}$ to octal

$0.AB_{16}$

$\rightarrow$ $0.1010\ \ 1011_2$

$\rightarrow$ 0.10101011_2

$\rightarrow$ $000.101\ \ 010\ \ 110_2$

$\rightarrow$ 0.526_8

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

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

Conversion between any two bases, other than decimal (Special cases)

- *Example 17*: Convert $AB.BA_{16}$ to octal

$AB.BA_{16}$

→ $1010\ 1011.1011\ 1010_2$

→ 10101011.1011101_2

→ $010\ 101\ 011.101\ 110\ 100_2$

→ 253.564_8

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

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

Conversion between any two bases, other than decimal (Special cases)

- *Example 18*: Convert 123_8 to hexadecimal

123_8

$\rightarrow 001\ \ 010\ \ 011_2$

$\rightarrow 1010011_2$

$\rightarrow 0101\ \ 0\,011_2$

$\rightarrow 53_{16}$

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

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

Conversion between any two bases, other than decimal (Special cases)

- *Example 19*: Convert 0.123_8 to hexadecimal

0.123_8

$\rightarrow$ $0.001\ 010\ 011_2$

$\rightarrow$ 0.001010011_2

$\rightarrow 0000.0010\ 1001\ 1000_2$

$\rightarrow 0.298_{16}$

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

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

Conversion between any two bases, other than decimal (Special cases)

- ***Example 20***: Convert 321.123_8 to hexadecimal

321.123_8

$\rightarrow$ $011\ 010\ 001.001\ 010\ 011_2$

$\rightarrow$ 11010001.001010011_2

$\rightarrow$ $1101\ 0001.0010\ 1001\ 1000_2$

$\rightarrow$ $D1.298_{16}$

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

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

Signed Numbers

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

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

In radix R systems,
the sign is encoded as:
MSD < R/2 → positive
MSD ≥ R/2→ negative

Sign and Magnitude

- *Example 21*: 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 22***: 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 23***: Convert –0.0A$_{16}$ to binary using ***sign and magnitude*** method

0.0A$_{16}$

➔ 0000.0000 1010$_2$

➔ 0.0000101$_2$

unsigned value

–0.0A$_{16}$

➔ 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

2's Complement

- In binary arithmetic, the ***two's complement*** of an ***N-bit*** number is formed by
  - *Subtraction from* 2^N.
    The ***two's complement*** of 01100101_2 is
    $100000000 - 01100101_2 = 10011011_2$
  - *Inverting its bits* and *adding 1*.
    The ***two's complement*** of 01100101_2 is
    $10011010_2 + 1 = 10011011_2$.
  - *Working from LSB towards MSB*
    start at the least significant bit (LSB), and copy all the zeros (working from LSB toward the most significant bit) until the first 1 is reached; then copy that 1, and flip all the remaining bits
    The ***two's complement*** of 01100101_2 is 10011011_2.

2's Complement

- ***Example 24***: 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 25***: 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
0000	**0**	**+0**	**+0**
0001	1	+1	+1
0010	2	+2	+2
0011	3	+3	+3
0100	4	+4	+4
0101	5	+5	+5
0110	6	+6	+6
0111	7	+7	+7
1000	8	**– 0**	**– 8**
1001	9	– 1	– 7
1010	10	– 2	– 6
1011	11	– 3	– 5
1100	12	– 4	– 4
1101	13	– 5	– 3
1110	14	– 6	– 2
1111	15	– 7	– 1

For a given *n* bit binary pattern

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

Unsigned

- ***Example 26***: 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:

$$
\begin{aligned}
11011.11011_2 &= 1101111011_2 / 100000_2 \\
&= 891_{10} / 32_{10} \\
&= 27.84375_{10}
\end{aligned}
$$

Sign and Magnitude

- ***Example 27***: Convert 11011.11011_2 to decimal, assuming that it is encoded using ***sign and magnitude*** method.

$11011.11011_2 \rightarrow -1011.11011_2$

$1011_2 \rightarrow 11_{10}$

$0.11011_2 \rightarrow 0.84375_{10}$

$1011.11011_2 \rightarrow 11.84375_{10}$

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

Another method:

$11011.11011_2 \rightarrow -1011.11011_2$

$1011.11011_2 = 101111011_2 / 100000_2$

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

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

2's Complement

- **Example 28**: Convert 11011.11011_2 to decimal, assuming that it is encoded using ***2's complement*** method.

$11011.11011_2 \rightarrow$ *negative number*

$11011.11011_2 \rightarrow -00100.00101_2$

$00100_2 \rightarrow 4_{10}$

$0.00101_2 \rightarrow 0.15625_{10}$

$00100.00101_2 \rightarrow 4.15625_{10}$

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

Another method:

$11011.11011_2 \rightarrow$ *negative number*

$11011.11011_2 \rightarrow -00100.00101_2$

$00100.00101_2 = 0010000101_2 / 100000_2$

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

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

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 29***: 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 30***: 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 31***: Convert **111111011**$_2$ to decimal, assuming that it is encoded using ***2's complement*** method.

- $111111011_2 \rightarrow$ *negative number*
- $111111011_2 \rightarrow -000000101_2$
- $000000101_2 \rightarrow 5_{10}$
- $111111011_2 \rightarrow -5_{10}$

2's Complement

- *Example 32*: Convert $-AB.BA_{16}$ to binary 2's complement
  Normalize your answer.

$AB.BA_{16}$

$\rightarrow 10101011.10111010_2$

$+AB.BA_{16}$

$\rightarrow 010101011.10111010_2$

$-AB.BA_{16}$

$\rightarrow 101010100.01000110_2$

- *After normalization*, $-AB.BA_{16}$

$\rightarrow 1.0101010001000110_2 \times 2^{+8}$

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 33***: Convert $-AB.BA_{16}$ to binary 2's complement
  Normalize your answer and
  - *limit it (using truncation) to 6 bits (1 + 5 bits) in total*
  - *limit it (using truncation) to 8 bits (1 + 7 bits) in total*
  - *limit it (using truncation) to 11 bits (1 + 10 bits) in total*
  - *limit it (using truncation) to 15 bits (1 + 14 bits) in total*

$AB.BA_{16} \rightarrow 10101011.10111010_2$

$+AB.BA_{16} \rightarrow 010101011.10111010_2$

$-AB.BA_{16} \rightarrow 101010100.01000110_2$

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

- *After normalization*, $-AB.BA_{16}$

  $\rightarrow 1.0101010001000110_2 \times 2^{+8}$
- *After limiting the answer to 6 bits (5+1) in total*,

  $\rightarrow 1.01010_2 \times 2^{+8}$ *(using truncation)*
- To read this number,

  $1.01010_2 \times 2^{+8} \rightarrow 101010000_2$

  This is a ***negative*** number.

  Its absolute value is $0\ 1011\ \ 0000_2 = B0_{16}$
- $1.01010_2 \times 2^{+8} \rightarrow -B0_{16}$
- Truncation error $= -AB.BA_{16} - (-B0_{16}) = 4.46_{16}$

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

- *After normalization*, $-AB.BA_{16}$

  $\rightarrow 1.0101010001000110_2 \times 2^{+8}$
- *After limiting the answer to 8 bits (1 + 7) in total*,

  $\rightarrow 1.0101010_2 \times 2^{+8}$ *(using truncation)*
- To read this number,

  $1.0101010_2 \times 2^{+8} \rightarrow 101010100_2$

  This is a ***negative*** number.

  Its absolute value is $0\ 1010\ 1100_2 = AC_{16}$
- $1.0101010_2 \times 2^{+8} \rightarrow -AC_{16}$
- Truncation error $= -AB.BA_{16} - (-AC_{16}) = 0.46_{16}$

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

- *After normalization*, –AB.BA$_{16}$
  ➔ $1.0101010001000110_2 \times 2^{+8}$
- *After limiting the answer to 11 bits (10+1) in total*,
  ➔ $1.0101010001_2 \times 2^{+8}$ *(using truncation)*
- To read this number,
  $1.0101010001_2 \times 2^{+8}$ ➔ 101010100.01_2
  This is a ***negative*** number.
  Its absolute value is 0 1010 1011.11$_2$
  0 1010 1011.11$_2$ ➔ 0 1010 1011.1100$_2$ =AB.C$_{16}$
- $1.0101010001_2 \times 2^{+8}$ ➔ –AB.C$_{16}$
- Truncation error = –AB.BA$_{16}$ – (–AB.C$_{16}$) = 0.06$_{16}$

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

- *After normalization*, $-AB.BA_{16}$

  $\rightarrow 1.0101010001000110_2 \times 2^{+8}$
- *After limiting the answer to 15 bits (14+1) in total*,

  $\rightarrow 1.01010100010001_2 \times 2^{+8}$ *(using truncation)*
- To read this number,

  $1.01010100010001_2 \times 2^{+8} \rightarrow 101010100.010001_2$

  This is a ***negative*** number.

  Its absolute value is $0\ 1010\ 1011.1011\ 11_2$

  $01010\ 1011.1011\ 11_2 \rightarrow 01010\ 1011.1011\ 1100_2$

  $=AB.BC_{16}$
- $1.01010100010001_2 \times 2^{+8} \rightarrow -AB.BC_{16}$
- Truncation error = $-AB.BA_{16} - (-AB.BC_{16}) = 0.02_{16}$

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