Binary Coded Decimal (BCD):

- Decimal numbers are more natural to humans. Binary numbers are natural to computers. Quite expensive to convert between the two.
- If little calculation is involved, we can use some *coding schemes* for decimal numbers.
- One such scheme is BCD, also known as the *8421* code.
- Represent each decimal digit as a 4-bit binary code.

Decimal digit **BCD**	0 **0000**	1 **0001**	2 **0010**	3 **0011**	4 **0100**
Decimal digit **BCD**	5 **0101**	6 **0110**	7 **0111**	8 **1000**	9 **1001**

Some codes are unused, eg: $(1010)_{BCD}$, $(1011)_{BCD}$, …, $(1111)_{BCD}$. These codes are considered as errors.

- Easy to convert, but arithmetic operations are more complicated.
- Suitable for interfaces such as keypad inputs and digital readouts.
- Examples:

$(234)_{10} = (0010\ 0011\ 0100)_{BCD}$

$(7093)_{10} = (0111\ 0000\ 1001\ 0011)_{BCD}$

$(1000\ 0110)_{BCD} = (86)_{10}$

$(1001\ 0100\ 0111\ 0010)_{BCD} = (9472)_{10}$

Notes: BCD is not equivalent to binary.

Example: $(234)_{10} = (11101010)_2$

Excess-3:

- Each decimal digit is taken, 3 is added to it and the resulting value is converted to its 4 bit binary value.
  Example:
  $(234)_{10} = (0101\ 0110\ 0111)_{excess\text{-}3}$

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

Unsigned number:

- Numbers don't hold any sign
- Range: 0 to 2^n-1 for n bits

Negative Numbers Representation:

- There are three common ways of representing signed numbers (positive and negative numbers) for binary numbers:
  - Sign-and-Magnitude
  - 1s Complement
  - 2s Complement

Sign-and-Magnitude:

- Negative numbers are usually written by writing a minus sign in front.
  - Example:

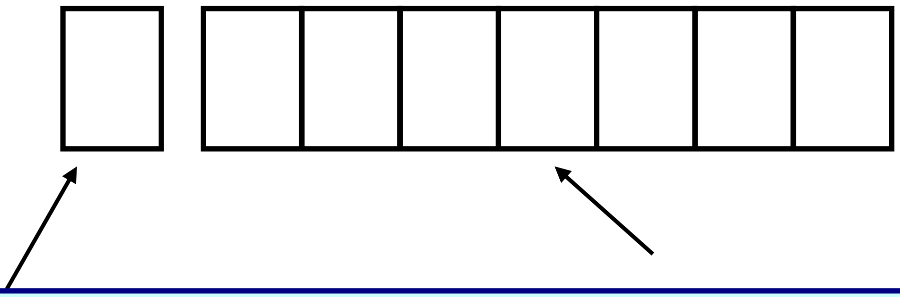
    $-(12)_{10}$, $-(1100)_2$

- In computer memory of fixed width, this sign is usually represented by a bit:

  0 for +

  1 for -

  Example: an 8-bit number can have 1-bit sign and 7-bits magnitude.



$1101_2 = 13_{10}$ (a 4-bit unsigned number)

0 $1101 = +13_{10}$ (a positive number in 5-bit signed magnitude)

1 $1101 = -13_{10}$ (a negative number in 5-bit signed magnitude)

$0100_2 = 4_{10}$ (a 4-bit unsigned number)

0 $0100 = +4_{10}$ (a positive number in 5-bit signed&magnitude)

1 $0100 = -4_{10}$ (a negative number in 5-bit signed magnitude)

$0100_2 = 4_{10}$ (

0 $0100 = +4_{10}$ (

1 $0100 = -4_{10}$ (a negative number in 5-bit signed magnitude)

- Largest Positive Number: 0 1111111 $+(127)_{10}$
- Largest Negative Number: 1 1111111 $-(127)_{10}$
- Zeroes: 0 0000000 $+(0)_{10}$

  1 0000000 $-(0)_{10}$
- Range: $-(127)_{10}$ to $+(127)_{10}$
- Signed numbers needed for negative numbers.

  Range: $=-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

1s Complement:

- Given a number *x* which can be expressed as an *n*-bit binary number (*i.e. integer part has n digits and fraction has m digit*), its negative value can be obtained in 1s-complement representation using:

+7	0111	-7	1000
+6	0110	-6	1001
		-5	1010
		-4	1011
		-3	1100
		-2	1101
		-1	1110

+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000

-12 in 1’s complement:

$$-(00001100)_2$$

invert the bits by replacing the 1s with 0s and 0s with 1s

$$(11110011)_2$$

- Essential technique: invert all the bits.

  Examples: 1s complement of 00000001 = $(11111110)_{1s}$

  1s complement of 01111111 = $(10000000)_{1s}$

- Largest Positive Number: 0 1111111 $+(127)_{10}$
- Largest Negative Number: 1 0000000 $-(127)_{10}$
- Zeroes: 0 0000000

  1 1111111

- Range: $-(127)_{10}$ to $+(127)_{10} = -(2^{n-1} - 1)$ to $+ (2^{n-1} - 1)$
- The most significant bit still represents the sign:

  0 = +ve; 1 = -ve.

- Examples (assuming 8-bit binary numbers):

  $(14)_{10} = (00001110)_2 = (00001110)_{1s}$

  $-(14)_{10} = -(00001110)_2 = (11110001)_{1s}$

2s Complement:

- Method 1: Essential technique: invert all the bits and add 1.
  Examples:

  2s complement of

  $(00000001)_{2s} = (11111110)_{1s}$ (invert i.e 1's complement)

  $= (11111111)_{2s}$ (add 1)

  2s complement of

  $(01111110)_{2s} = (10000001)_{1s}$ (invert i.e 1's complement)

  $= (10000010)_{2s}$ (add 1)

- Method 2: Keep unchanged till 1st occurrence of 1 from LSB and invert remaining 1's into 0's and 0's into 1's till MSB

$(01111110)_{2s} = (10000010)_{2s}$

- Largest Positive Number: 0 1111111 $+(127)_{10}$
- Largest Negative Number: 1 0000000
  $-(128)_{10}$
- Zero: 0 0000000
- Range: $-(128)_{10}$ to $+(127)_{10} = -(2^{n-1})$ to $+(2^{n-1}-1)$
- The most significant bit still represents the sign:

  0 = +ve; 1 = -ve.
- Examples (assuming 8-bit binary numbers):

  $(14)_{10} = (00001110)_2 = (00001110)_{2s}$

  $-(14)_{10} = -(00001110)_2 = (11110010)_{2s}$