



# Number system and conversions (Section 1.4 of textbook)

Vikas Dhiman for ECE275

August 26, 2023

## 1 Place value number system

- What is a place value number system?
- What are some examples?
- What are some non-examples?
- What is a radix (or base)?
- How to convert between different radix in place value system?
- What are some commonly used number systems in computer engineering? *Octal Hexadecimal*

# Place value system

## 1.1 Decimal number system

$$(1375)_{10} = 1 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$$

radix  
on  
the base  
= 10

↑  
10's place

Digit	1	3	7	5
Place value	$10^3$	$10^2$	$10^1$	$10^0$
Value	1000	300	70	5
	1375			

## 1.2 Binary numbers

$$(11101)_2$$

radix  
on the base  
= 2

Bit	1	1	1	0	1
Place value	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Value	16	8	4	0	1
	29				

Create a new number system with radix 7  
(354)<sub>7</sub> =

## 1.3 Conversion between different radix

Problem 1 Convert the following binary numbers to decimal:  $(11110)_2$ ,  $(100111)_2$ .

Bit	1	1	1	1	0
Place value	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
Value	16	8	4	2	0

= 30

	3	5	4
PV	$7^2$	$7^1$	$7^0$
V	147	35	4
	186		

Contd on Page 1

**Conversion from decimal to binary** The value is in decimal because we find it easy to do calculations in decimal numbers. Decimal values can be converted back to Binary representation by *repeated division* by 2 while noting down the remainder. Allow me to use / sign to denote both quotient and remainder after division. Let's convert  $(22)_{10}$  back to binary:

Contd on Page 1

$$(354)_7 = (186)_{10}$$

$$(d_n d_{n-1} \dots d_2 d_1 d_0)_r = d_n r^n + \dots + d_1 r^1 + d_0 r^0$$

$$= \sum_{i=0}^n d_i r^i$$

Converts a number from any radix  $r$  to its char<sup>n</sup> value

char<sup>n</sup> value  $\xrightarrow{\checkmark}$  int  $\xrightarrow{\quad}$  char<sup>n</sup>

"354"<sub>7</sub>  $\xrightarrow{\checkmark}$  186  $\xrightarrow{\quad}$  "186"<sub>10</sub>

representation of a number is a string  $\xrightarrow{\quad}$  " " <sub>5</sub>

Decimal to binary

123  $\rightarrow$  ( )<sub>2</sub>

2	123	
2	61	1
2	30	1
2	15	0
2	7	1
2	3	1
2	1	1
	0	1

123 = (1111011)<sub>2</sub>

$$89 = (1011001)_2$$

$$89 = (155)_7$$

1	5	5
$7^2$	$7^1$	$7^0$
$49 \times$	$35 \times$	5

7	89	
7	12	5
	1	5
	0	1

representation

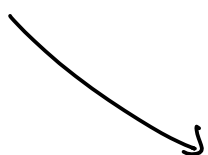
$(153)_7$



(

representation

)<sub>2</sub>



Value  
(Decimal)



$22/2 = (11, 0)$	11 is the quotient and 0 is the remainder
$11/2 = (5, 1)$	5 is the quotient and 1 is the remainder
$5/2 = (2, 1)$	
$2/2 = (1, 0)$	
$1/2 = (0, 1)$	

Read the remainders from bottom to top and right them as left to right, to form the resultant binary number  $(22)_{10} = (10110)_2$ .

**Problem 2** Find the binary representation for decimal numbers: 123 and 89. Show your work.

## 2 Hexadecimal numbers

Numbers with base 16 are called Hexadecimal numbers. From 0 to 9 the symbols are same as decimal numbers. From 10 to 15, Hexadecimal numbers use A to F.

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

. Example,  $(10AD)_{16} = 1 \times 16^3 + 10 \times 16^1 + 13 = 4096 + 160 + 13 = 4269$ .

## 3 Octal numbers

Numbers with base 8 are called octal numbers. Example,  $(354)_8 = 3 \times 8^2 + 5 \times 8 + 4 = 192 + 40 + 4 = 236$ .

## 4 Hexadecimal/octal to binary and vice-versa

Normally, if you have to convert between a number of base  $r_1$  to a number of base  $r_2$ , we will have to convert it via decimal numbers. Convert from base  $r_1$  to decimal and then from decimal to  $r_2$ .

Since Hexadecimal base 16 is an exact power of 2 ( $16 = 2^4$ ). Conversion between Hexadecimal to binary is easy. You can group 4 binary digits from right to left and convert each group of 4 binary digits to a single Hexadecimal digit and back. Example,  $(10110)_2 = (0001.0110)_2 = (16)_{16}$ . To convert back. Take example,  $(10AD)_{16} = (0001.0000.1010.1101)_2 = (1.0000.1010.1101)_2$ .

**Problem 3** Find the binary and decimal values of the following Hexadecimal numbers  $(A25F)_{16}$ ,  $(F0F0)_{16}$ .

Hexadecimal  
Octal

$$\text{radix} = 16 = 2^4$$

$$\text{radix} = 8 = 2^3$$

Decimal

Binary	Value	Symbol
0000	0	0
0001	1	
0010	2	
0011	3	
0100	4	
0101	5	
0110	6	
0111	7	
1000	8	
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

$$(ABC)_{16} \rightarrow (2748)_{10}$$

Hex digit	A <sub>(10)</sub>	B <sub>(11)</sub>	C <sub>(12)</sub>
Place value	$16^2$	$16^1$	$16^0$
Value	2560	176	12

2748

Hexadecimal  $\longleftrightarrow$  Binary (2)

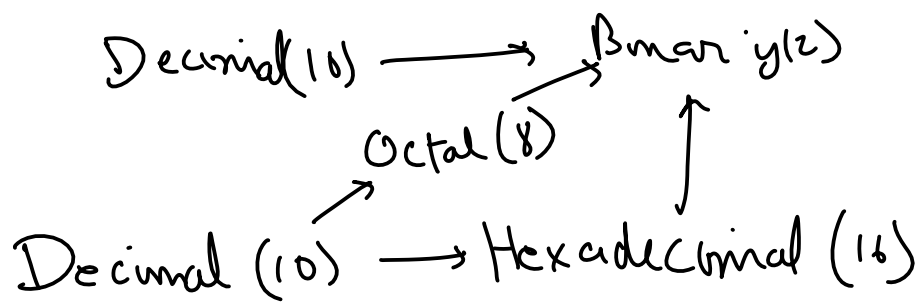
$$(3AA)_{16} \quad (2^4)$$

$$\left( \underbrace{11}_3 \underbrace{1010}_A \underbrace{1010}_A \right)_2$$

$$(A7C)_{16} \rightarrow (1010 \ 0111 \ 1100)_2$$

Octal (2<sup>3</sup>)  $\longleftrightarrow$  Binary (2)

$$(567)_8 \rightarrow (101 \ 110 \ 111)_2$$



2	1137
2	568
	1

16	1137
16	71
	47

$$(1137)_{10} \rightarrow (471)_{16} \rightarrow (100\ 0111\ 0001)_2$$

Similarly octal to binary can proceed by grouping 3-binary digits at a time. Example,  $(354)_8 = (011\ 101\ 100)_2$ .

**Problem 4** Find the binary and decimal values of the following Octal numbers  $(3751)_8$  and  $(722)_8$ .

## 5 Signed binary numbers

Signed numbers include both negative and positive numbers. There three common signed number representations

1. Sign magnitude representation
2. One's complement
3. Two's complement

### 5.1 Sign-magnitude representation

The Most significant (left most) *bit* (binary digit) represents sign ( $0 = +$  and  $1 = -$ ), the rest represent the magnitude. Example, a 5-bit number  $(11010)_2$  in signed magnitude representation has the value of  $(-1010)_2 = -10$ . Note that  $+10$  has to be represented by a leading 0 at the most significant bit (MSB)  $+10 = (01010)_2$ . Hence, the number of bits have to be specified.

**Problem 5** • Write down all possible 4-digit binary numbers and corresponding decimal values if they are in signed magnitude format? What is the minimum and maximum value?

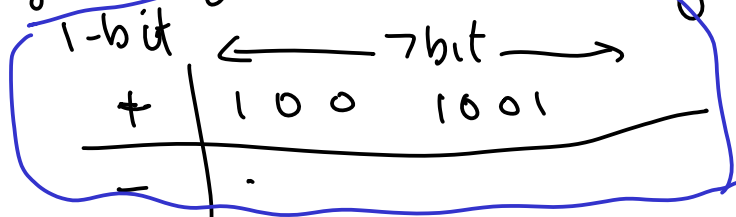
- What is the minimum and maximum value of  $n$ -digit signed binary number in sign-magnitude format?

Signed decimal numbers  
 $\begin{matrix} +ve & -ve \\ 1137 & -1137 \end{matrix}$

$$2^0 - 2^{10}$$

$$2^{10} \approx 1024 \approx 1000$$

Sign-magnitude binary numbers (8-bit)



$$2^8 = 256$$

1-bit for the sign

$$\begin{matrix} + & \equiv & 0 \\ - & \equiv & 1 \end{matrix}$$

$$1111\ 1111$$

Unsigned binary number (8-bits)  $0 - 255 = 0 - (2^8 - 1)$



Unsigned decimal number (8-digits) 0-9999 9999

$$0 - (10^8 - 1)$$

What is the range of 8-bit unsigned signed binary numbers in sign-magnitude notation.

1-bit for sign

128 +ve 0-127

128 in the negative direction  
-0 - (-127)

+ 000 0000  
- 000 0000  
magnitude

Range is from

-127 to +127

$$-(2^7 - 1) \text{ to } +(2^7 - 1)$$

$$x + y$$

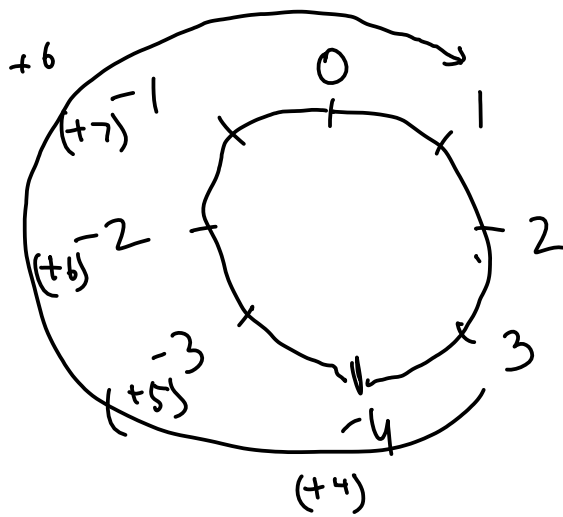
$$x - y \approx x + (-y)$$

$$-(-y) = +y$$

Desired properties of -ve numbers

2's complement satisfies these properties  
4-bit





$$3-2 = 3+(-2) = 1$$

If you allow  
for looping  
around then  
you can get  
signed arithmetic  
for free

2's complement notation

4-bit number

$$-n = (2^4 - n)$$

$$\begin{aligned} -2_{10} &= -(0010)_2 = (110000 - 0010)_2 \\ &= (1110)_2 \end{aligned}$$

$$(0011)_2 = 3$$

$$(1110)_2 = -2$$

---


$$(0001)_2 = 1$$

2's complement notation

+ve numbers stay the same  
4-bit signed binary number

$$0 - (2^3 - 1) = 0 - 7$$

same binary notation

0000		0
		⋮
0111		7

-ve numbers  $-n = (2^4 - n)$

Decimal	2's complement binary
-8	1000
-7	1001
-6	1010
-5	(1011) <sub>2</sub>
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

2's complement negation shortcut

(a) Flip all the bits

(b) Add 1

$$-5_{10} = -(0101)_2$$

$$= (1010)_2 + 1$$

$$= (1011)_2$$

$$-8_{10} = -(1000)$$

$$-2_{10} = -(0010)_2$$

$$= (1101)_2 + 1$$

$$= (1110)_2$$

1's complement notation

4-bit

$$-n = (2^4 - 1) - n = (1111)_2 - n$$

$$-3_{10} = -(0011)_2$$

$$= (1100)_2$$

flip the bits

---

- 8

,

,

,

in

1's complement  
notation

+ 7

## 5.2 One's complement negation

You can convert a positive number (say  $+10$ ) to negative number by applying a negative sign in front of it ( $-(+10) = -10$ ). It is more evident from taking negative of a negative number ( $-(-10) = +10$ ). In case of sign-magnitude representation, the “negative operator” flips the sign bit. The next two signed number representations (1's complement and 2's complement) are designed around specific negative operator definitions.

Negate  $13_{10} = 01101_2$  using 5-bit one's complement.

Negate  $-13_{10}$  using 5-bit one's complement.

## 5.3 One's complement binary numbers

In one's complement representation, the negative operation is obtained by flipping all the bits of the binary number. Example, a 5-bit one's complement of  $+10 = (01010)_2$  is  $(10101)_2 = -10$ . Note that flipping bits is equivalent to subtracting the number from  $(11111)_2$ , hence the name. You can also confirm that double negative operator yields back the same number.

- Problem 6**
- Write down all possible 4-digit binary numbers and corresponding decimal values if they are in sign magnitude format? What is the minimum and maximum value?
  - What is the minimum and maximum value of  $n$ -digit signed binary number in one's complement?

**Problem 7** Determine the decimal values of the following 1's complement 6-digit binary numbers :

1. 01101110

2. 10101101

**Problem 8** Convert the decimal numbers -17 and +23 into the 6-digit one's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.

## 5.4 Two's complement negation

In two's complement representation, the n-digit negative number is obtained by subtracting the positive number from  $2^n$ . Example, two's complement of 5-digit binary number  $+10 = (01010)_2$  is  $2^5 - 10 = 22 = (11000)_2$ . An easier algorithm to get two's complement goes via one's complement. Note that  $(11111)_2 = 2^5 - 1$ . We can get two's complement by adding 1 to one's complement. To get two's complement:

1. Flip all the bits. (Same as taking one's complement).
2. Add 1 to the number.

Negate  $13_{10} = 01101_2$  using 5-bit two's complement.

Negate  $-13_{10}$  using 5-bit two's complement.

How to convert one's complement number representation into sign-magnitude numbers?

1. Check if the number is positive or negative. Even for one's complement representation, or two's complement representation, if the MSB (Most-significant bit) is 1, then the number is negative, otherwise positive.
2. If positive: For positive numbers, two's complement, one's complement and sign magnitude are the same. No conversion between different representation is needed. 2.b If negative: For negative numbers. Flip the bits of 1's complement. Once you flip the 1's complement bits of a negative number, you get the corresponding positive number.
3. We still want to represent the original negative number. So we set the MSB of sign-magnitude representation to 1. Since the range (min and max) for both n-bit 1's complement and sign-magnitude are the same (between  $-(2^{n-1} - 1)$  and  $2^{n-1} - 1$ ), you can always represent 8-bit 1's complement numbers with needing to extend the 8-bit number to 9-bits.

Example: Convert 8-bit one's complement 10101010 to 8-bit sign-magnitude Let number n = 10101010

1. Is the number +ve or -ve: It is negative because it starts with 1.
2. The number is not positive.
3. Take the 1's complement of the negative number to get the positive part. i.e. Flip the bits:  
 $-n = 01010101$  or  $n = -(01010101)$
4. We got the positive part of the number, but we want to represent the original negative number, so we set the MSB bit one. Hence, the equivalent sign-magnitude representation is:  
 $n = 11010101$

## 5.5 Two's complement representation

**Problem 9** *Determine the decimal values of the following 2's complement 6-digit numbers :*

1. *01011110*

2. *10010111*

**Problem 10** *Convert the decimal numbers -17 and +23 into the 6-digit two's complement binary numbers and try adding them. What adjustments will you need to make to get the right result's (23-17=6) in binary representation.*



**Problem 11** *Convert the decimal numbers 73, 23, -17, and -163 into signed 8-bit numbers in the following representations:*

1. *Sign and magnitude*
2. *1's complement*
3. *2's complement*

## 5.6 Arithmetic overflow

**Problem 12** *Consider addition of 4-digit two's complement binary numbers*

1.  $1010_2 + 1101_2$
2.  $1011_2 + 1100_2$

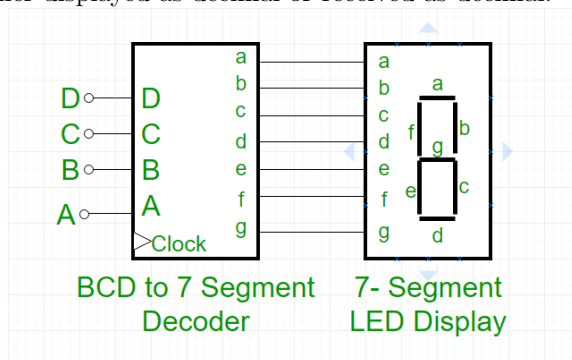
*In which of the two case overflow happens? Can you come up with a rule to “easily” detect overflow?*

### 5.6.1 Rules for detecting arithmetic overflow:

1. Adding numbers of different signs never produces an overflow.
2. Adding numbers of the same sign may produce an overflow
  - (a) Wrong approach: Adding two negative 2's complement numbers always produces an additional carry-over 1, but that in itself isn't an overflow. An example, the range of 4-bit 2's complement numbers is between -8 to +7. Adding -3 to -4 in 2's complement is  $1101 + 1100$  produces an additional carry over 1. You can ignore the additional carry-over 1 to get the correct answer  $1001 = -7$  which is within range -8 to 7.
  - (b) Approach 1: The easiest way for now to detect overflow is if adding two -ve numbers results in a +ve number, or adding +ve numbers results in a -ve number.
  - (c) Approach 2: You can also do a range test in decimal based range test. The range of n-bit 2's complement numbers is between  $-2^{n-1}$  and  $2^{n-1} - 1$ . For 5-bit 2's complement numbers, it is between -16 and 15. For 6-bit 2's complement numbers, it is between -32 and 31.
  - (d) Approach 3: You can also check the carry-overs of the most significant two bits. If they match, i.e. 0 and 0, or 1 and 1, then there is no overflow. If they do not match, i.e. 0 and 1 or 1 and 0, then there is an overflow.

## 6 Binary coded decimal

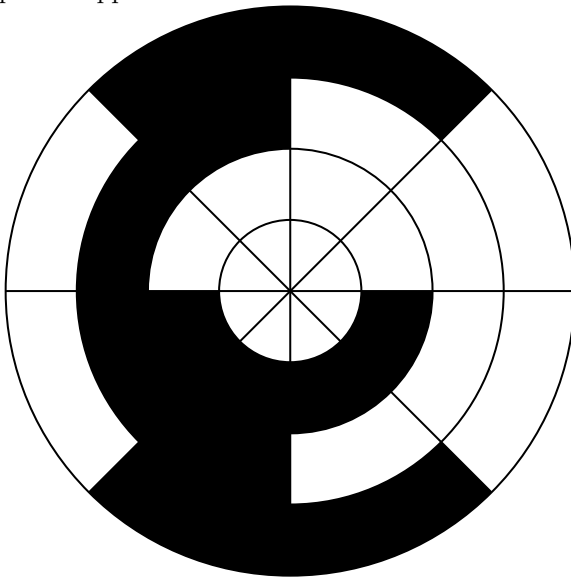
In Binary coded decimal (BCD), each decimal digit is represented by 4 bits. For example,  $1047 = (0001_0000_0100_0111)_{BCD}$ . It is useful in input-output applications where the number has to be either displayed as decimal or received as decimal.



**Problem 13** Convert 11, 23, 35, 57 and 103897 to BCD?

## 7 Gray code

A sequence of binary numbers where only one bit changes when the number increases by 1. It is helpful in applications like wheel encoders



**Problem 14** *Write all possible 3-bit binary numbers in gray-code*