Chapter 1

Number Systems And Codes

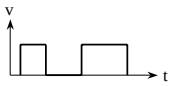
Computer Application

Chapter Outline

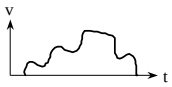
- Analog vs. Digital
- Positional Number Systems
- Positional Number System Conversions
- Binary System operations
- Encoding Techniques

Digital vs. Analog

- Digital characteristics
  - Discrete signal levels (voltage usually)
  - Two levels: on/off, high/low 1/0 (binary)
  - Disjoint or quantized level changes



- Analog characteristics
  - Continuous signal levels
  - Very small, smooth level changes



Digital vs. Analog

- Natural world is analog
- Many devices better when digital:

Digital	**Analog**
Compact Disc	Magnetic Tape
Microcomputer-controlled Engine	Mechanically-controlled Engine
Telephone System	Telephone System
Movie Special Effects	Movie Special Effects
Digital Computers:	Analog Computer:
PC, Mainframe, Supercomputer	OpAmp, Res, Cap

Digital vs. Analog

- Advantages of each technology:

Digital	**Analog**
Reproducible results	Less complex ?
Ease of design	
Flexible and function	Higher speed ?
Programmable	
High speed	
Economical	

Positional Number System

- General form of a number :

$$d_{p-1}d_{p-2}\ldots d_1 d_0 . d_{-1}d_{-2}\ldots d_{-n}$$

- The value of the number :

$$D = \sum_{i=-n}^{p-1} d_i \cdot r^i$$

- **. : radix point**
- ***r* : *base* or *radix***
- ***dp-1* : the *most significant digit***
- ***d-n* : the *least significant digit***

Decimal System

- ***d* : 0 , 1 , 2 , . . . , 9**
- ***r* = 10**
- **Example : 2081.35**
  ***p* = 4 , *n* = 2**

$d_3 = 2, d_2 = 0, d_1 = 8, d_0 = 1, d_{-1} = 3, d_{-2} = 5$

$$2081.35 = \sum_{i=-2}^{3} d_i \cdot 10^i$$

$$2081.35 = 2\cdot 10^3 + 0\cdot 10^2 + 8\cdot 10^1 + 1\cdot 10^0 + 3\cdot 10^{-1} + 5\cdot 10^{-2}$$

$$2081.35 = 2\cdot 1000 + 0\cdot 100 + 8\cdot 10 + 1\cdot 1 + 3\cdot 0.1 + 5\cdot 0.01$$

Binary System

The form of a binary number is:

$$b_{p-1}b_{p-2}\ldots b_1 b_0 . b_{-1}b_{-2}\ldots b_{-n}$$

The decimal value of the number is:

$$B = \sum_{i=-n}^{p-1} b_i \cdot r^i$$

- ***r* = 2 (binary radix)**
- **. : binary point**
- ***bi* (binary digit = *bit*) : 0 , 1**
- ***bp-1* : the *most significant bit (MSB)***
- ***b-n* : the *least significant bit (LSB)***

Example :

- **11101.01**
- ***p* = 5 , *n* = 2**

$b_4 = 1, b_3 = 1, b_2 = 1, b_1 = 0, b_0 = 1, b_{-1} = 0, b_{-2} = 1$

$$B = \sum_{i=-2}^{4} b_i \cdot 2^i$$

$$B = 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 0\cdot 2^{-1} + 1\cdot 2^{-2}$$

$$B = 16 + 8 + 4 + 0 + 1 + 0 + 0.25 = 29.25$$

Exercise

- **Calculate the equivalent decimal numbers :**
  1 , 10 , 11 , 100 , 101, 10101.11
- **Answer :**
  1 (2) = 1 (10)
  10 (2) = 2 (10)
  11 (2) = 3 (10)
  100 (2) = 4 (10)
  101 (2) = 5 (10)
  10101.11 (2) = 21.75 (10)

Octal and Hexadecimal Numbers

- **Octal number System :**
  - *r* = 8
  - *d* : 0 , 1 , 2 , . . . , 7
- **Hexadecimal number System :**
  - *r* = 16
  - *d* : 0 , 1 , 2 , . . . , 9, A, B, C, D, E, F
- **The radices are powers of 2**
- **Used for shorthand representations of long binary numbers**

Binary, Decimal, Octal and Hexadecimal Numbers

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Binary - Octal/Hexadecimal Conversion

- **<u>Binary to Octal/Hexadecimal :</u>**
  Starting from the decimal point :
  - Separate the bits into groups of three/four
  - Replace each group with its corresponding Octal/Hexadecimal digit
- **Examples :**
  - **100011110.10101$_{(2)}$ = 100 011 110.101 010 = 436.52$_{(8)}$**
  - **1011101000.001111$_{(2)}$ = 0010 1110 1000.0011 1100 = 2E8.3C$_{(16)}$**

Binary - Octal/Hexadecimal Conversion

- **<u>Octal/Hexadecimal to Binary:</u>**
  Convert each Octal/Hexadecimal digit into its corresponding 3 / 4 bit string
- **Examples :**
  - **621.3(8) = 110010001.011(2)**
  - **F5A.2C(16) = 111101011010.00101100(2)**

Exercise

- **Convert 101100.101 into hexadecimal, octal and decimal.**
- **Convert F4A into binary**
- **Answers :**
  - **101100.101 $_{(2)}$ = 2C.A $_{(16)}$ = 44.625 $_{(10)}$**
  - **101100.101 $_{(2)}$ = 54.5 $_{(8)}$**
  - **F4A $_{(16)}$=111101001010 $_{(2)}$**

General Positional Number Conversion

- ***radix-r* to decimal :**

$$D = \sum_{i=-n}^{p-1} d_i \cdot r^i$$

- **decimal to *radix-r* :**
  - Successive division of D by *r*
  - The remainder of the long divsion will give the digits starting from the *least significant digit*

Example - Decimal to Binary :

- 179 $_{(10)}$
  179/2 = 89 (1) LSB
  89/2 = 44 (1)
  44/2=22 (0)
  22/2=11 (0)
  11/2=5 (1)
  5/2=2 (1)
  2/2=1 (0)
  1/2=0 (1)MSB
- Result : **10110011 $_{(2)}$**

Exercise

- Convert 154 into binary.
- Answer : **10011010**

Example - Decimal to Binary (Fraction):

- .375 (10)
  .375 * 2 = 0.75 (0)
  .75 * 2 = 1.5 (1)
  .5 * 2 = 1.0 (1)
  Fin
- Result : **.011 (2)**

Binary Addition

Binary addition table :

carry(in)	x	y	x+y	carry(out)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Example :

```
      1 0 0     0 1 0 1 1 0 0 0
x     1 7 3     1 0 1 0 1 1 0 1
y   + 4 4       0 0 1 0 1 1 0 0
      -----     ---------------
      2 1 7     1 1 0 1 1 0 0 1
```


Exercise

- **1 1 0 1 0 + 0 1 1 1 1 = 1 0 1 0 0 1**
  26+15 = 41
- **0 1 0 1 0 1 + 0 1 1 0 0 1 = 1 0 1 1 1 0**
  21+25=46

Representation of Natural Numbers in Binary Systems (Unsigned Number)

- There has 2^n parmutation in order n bits
- The range for n-bit is :

$$\boxed{\textit{from } 0 \textit{ to } +(2^n - 1)}$$

Binary Subtraction

- **Binary Subtraction table :**

borrow(in)	x	y	x-y	borrow(out)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Example :

```
      0 1 0     1 1 0 1 1 0 1 0
x     2 1 0     1 1 0 1 0 0 1 0
y   - 1 0 9     0 1 1 0 1 1 0 1
      -------   -----------------
      1 0 1     0 1 1 0 0 1 0 1
```


Exercise

- **1 1 0 1 0 - 0 1 1 1 1 = 0 1 0 1 1**
  26 - 15 = 11
- **1 1 1 0 1 - 1 0 1 1 1 = 0 0 1 1 0**
  29 - 23 = 6

Representation of Negative Numbers in Binary Systems

- **Signed-magnitude Representation.**
- **Two's-Complement Representation.**
- **One's-Complement Representation.**

Signed Magnitude Representation

- The MSB represents the sign bit (0 = +ve , 1 = -ve)
- The range for n-bit is :
- $\boxed{\textit{from } -(2^{n-1} - 1) \textit{ to } +(2^{n-1} - 1)}$
- Disadvantages :
  1- Two possible representations of zero
  2- Complicated digital adders

Example: n=5
* Range : from -15 to 15
* 10011= -3 , 01100 = +12
* 00000= 0 , 10000 = - 0

Two's Complement Representation

- The MSB represents the sign bit (0 = +ve , 1 = -ve)
- To calculate the value of negative number :
  1- Complement all bits of the positive number (one's complement)
  2- Add 1
- The range for n-bit is :

$$\boxed{\textit{from } -(2^{n-1}) \textit{ to } +(2^{n-1} - 1)}$$

- Advantages :
  1- Only one zero
  2- Addition and subtraction can be performed directly
- Disadvantage : One extra negative number (not symmetric)

Two's Complement Example

- n= 8
  - range from -128(10000000) to 127 (01111111)

$+100_{(10)}$ = 01100100
10011011
1 +
10011100 = $-100_{(10)}$

$0_{(10)}$ = 00000000
11111111
1 +
1 00000000 = $0_{(10)}$

One's Complement Representation

- The MSB represents the sign bit (0 = +ve , 1 = -ve)
- To calculate the negative number, complement all bits of the positive number
- The range for n-bit is :

$$\boxed{\textit{from } -(2^{n-1} - 1) \textit{ to } +(2^{n-1} - 1)}$$

- Advantages :
  1- Symmetry
  2- The ease of complementation
- Disadvantages:
  1- Two possible representations of zero
  2- Complicated digital Adders

One's-Complement Example

- n= 8
  range form -127(10000000) to 127 (01111111)
- $+100_{(10)}$ = 01100100

  $-100_{(10)}$ = 10011011
- $0_{(10)}$ = 00000000

  $-0_{(10)}$ = 11111111

Comparison (n=4)

Decimal	Signed Magnitude	One's Complement	Two's Complement
-8	-	-	1000
-7	1111	1000	1001
-6	1110	1001	1010
-5	1101	1010	1011
-4	1100	1011	1100
-3	1011	1100	1101
-2	1010	1101	1110
-1	1001	1110	1111
0	0000 or 1000	0000 or 1111	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

Exercise

- What is the representation of +21, - 21 in :
  6-bit signed magnitude representation
  6-bit one's complement
  6-bit two's complement
- Answer :
  +21= 010101 in the three represntations
  -21 = 110101 in the **signed magnitude representation**
  -21 = 101010 in **one's complement**
  -21 = 101011 in **two's complement**

Two's Complement Addition

- Addition rules (A+B)
  1- Use binary addition rules.
  2- Ignore any carry beyond the sign bit
- If the range is not exceeded addition result will be correct including the sign bit .

Examples:

```
  (-2)   1 1 1 0          (-3)   1 1 0 1
+ (-4)   1 1 0 0        (+3)   0 0 1 1
         ------                 ------
(-6) 1 1 0 1 0          (0)  1 0 0 0 0
```


Addition overflow

- Overflow detection rule :
  - The sign bit of the sum is different from the sign bit of the two addends
  Or,
  - The carry in(Cin) and the carry out (Cout) of the sign bit are different

Example :

```
      0 1 0 0 0          1 0 0 0 0
  (4)  0 1 0 0      (-4)  1 1 0 0
+(5)   0 1 0 1    + (-5)  1 0 1 1
       -------          -------
       1 0 0 1          1 0 1 1 1
```


Two's Complement Subtraction

- Method 1 : [A - B]
  1- Use binary Subtraction rules
  2- Ignore any borrow beyond the sign bit
- Example :

```
 (2)     0 0 1 0
-(4)     0 1 0 0
         -------
(-2)     1 1 1 0
```


Two's Complement Subtraction

- Method 2 : [A + (-B)]
  Add A to the Two's complement of B :
  1- Take the Ones's complement of B
  2- Add it to A with initial carry =1
- Example (2 - 4)

```
        1
  0 0 1 0 (2)
  1 0 1 1 (One's Complement of 4 )
  -------
  1 1 1 0  (-2)
```

- Overflow detection rule :
  - The sign bit of the result is different from the sign bit of A and the two's complement of B

Exercise

- Perform the following operations in 7 bit two's complement arithmetic :
  35+42, 26-42, -12-56

- 35 : 0100011 26 : 0011010
  +42 : 0101010 - 42 : 0101010
  ---------- ----------
  -51 1001101 (Overflow) -16 1110000

Exercise

- -12-56= -12+(-56) = -12+(Two's complement of 56)

  -12 : 1110100 , 56 : 0111000 , -56 : 1000111+1=1001000

- 1110100
  + 1001000

  0111100 (overflow)

Coding

- **Coding** : Representing a set of objects by a set of strings.
- **Code** : The set of bit strings.
- **Code Word** : A particular bit string in the **Code.**
- **Examples :**
  1- Data Objects : Decimal Numbers, Characters.
  2- Nondata Objects : Machine states, Control Actions.

Binary Codes For Decimal Numbers

- To represent the 10 decimal digits we need a minimum of 4 bits.
- **Examples :**
  - 4 bits Codes
    1- BCD (8421)- Binary Coded Decimal
    2- 2421
    3- Excess-3
  - Other Decimal codes
    1- Biquinary (7 bits)
    2- 1-out-of-10 (10 bits)

Decimal Codes (Table)

Decimal	BCD(8421)	2421	Excess-3	Biquinary	1-out-of-10
0	**0000**	**0000**	**0011**	**0100001**	**1000000000**
1	**0001**	**0001**	**0100**	**0100010**	**0100000000**
2	**0010**	**0010**	**0101**	**0100100**	**0010000000**
3	**0011**	**0011**	**0110**	**0101000**	**0001000000**
4	**0100**	**0100**	**0111**	**0110000**	**0000100000**
5	**0101**	**1011**	**1000**	**1000001**	**0000010000**
6	**0110**	**1100**	**1001**	**1000010**	**0000001000**
7	**0111**	**1101**	**1010**	**1000100**	**0000000100**
8	**1000**	**1110**	**1011**	**1001000**	**0000000010**
9	**1001**	**1111**	**1100**	**1010000**	**0000000001**

Exercise

- Write the BCD code, 2421 code, and Excess-3 code for 369
- Answer :
  BCD: **0011 0110 1001**
  2421: **0011 1100 1111**
  Excess 3 : **0110 1001 1100**

Gray Code

- One bit changes between two successive code words
- Binary Code and Gray Code (n = 3) :

Decimal	Binary Code	Gray Code
0	**000**	**000**
1	**001**	**001**
2	**010**	**011**
3	**011**	**010**
4	**100**	**110**
5	**101**	**111**
6	**110**	**101**
7	**111**	**100**

- Application :
  - Electromechanical applications - The Mechanical Encoding disk

Character Codes

- Each character is represented by a bit string
- **ASCII** - the American Standard Code for Information Interchange.
  - * Characters are encoded into 7 bits strings
  - * Contains 128 characters :
    - uppercase and lowercase letters
    - numbers
    - punctuation marks
    - control characters

ASCII Table

	$b_6b_5b_4$						
	000	001	010	011	100	101	110	111
$b_3b_2b_1b_0$								
0000			SP	0	@		‘	
0001			!	1	A		**a**	
0010			“	2	B		b	
0011			#	3	C		c	
⋮								
1111								

The Code for **a** is 1100001

Summary

- Digital Systems manipulate data in binary form.
- Arithmetic operations can be performed on Binary numbers.
- Hexadecimal numbers are used for shorthand representation of binary numbers.
- Arithmetic operations rules can be performed directly on negative numbers represented in Two’s complement.
- Decimal numbers , characters, actions, states can be encoded into binary strings.