Unit 1

Number Systems, Operations and Codes

5/20/2021

Devendra Chapagain

Unit 1: Number systems, operations and codes

Introduction to number system

A set of values used to represent quantities is called Number system. The number system uses different symbols and mathematical notations for representing numbers. The same sequence of symbols may represent different numbers in different numeral systems. For example, "11" represents the number eleven in the decimal numeral system (used in common life) and the number three in the binary numeral system (used in computers).

The common types of number system are:

1. Decimal Number system
   Base 10. Digits used: 0 to 9
2. Binary Number system
   Base 2. Digits used: 0, 1
3. Octal Number system
   Base 8. Digits used: 0 to 7
4. Hexadecimal Number system
   Base 16. Digits used: 0 to 9, Letters used: A- F

Conversion from one Number system to another

Decimal to Other Base System

Steps:

Step 1 – Divide the decimal number to be converted by the value of the new base.
Step 2 – Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
Step 3 – Divide the quotient of the previous divide by the new base.
Step 4 – Record the remainder from Step 3 as the next digit (to the left) of the new base number.
Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.
The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number.

Example: Convert $(29)_{10}$ into binary

Step	Operation	Result	Remainder
Step 1	29 / 2	14	1
Step 2	14 / 2	7	0
Step 3	7/2	3	1
Step 4	3/2	1	1
Step 5	½	0	1

As mentioned in Steps 2 and 4, the remainders have to be arranged in the reverse order so that the first remainder becomes the Least Significant Digit (LSD) and the last remainder becomes the Most Significant Digit (MSD).

Therefore: $29_{10} = 11101_2$

Other Base System to Decimal System

Steps

Step 1 – Determine the column (positional) value of each digit (this depends on the position of the digit and the base of the number system).
Step 2 – Multiply the obtained column values (in Step 1) by the digits in the corresponding columns.
Step 3 – Sum the products calculated in Step 2. The total is the equivalent value in decimal.
Example: Convert $(11101)_2$ into Decimal

Step	Binary Number	Decimal Number
Step 1	11101_2	$((1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2		$(16 + 8 + 4 + 0 + 1)_{10}$
Step 3		29

Therefore: $(11101)_2 = (29)_{10}$

Complements of Numbers

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulations. There are two types of complements for each base r- system.

1. r's complement
2. (r-1)'s complement

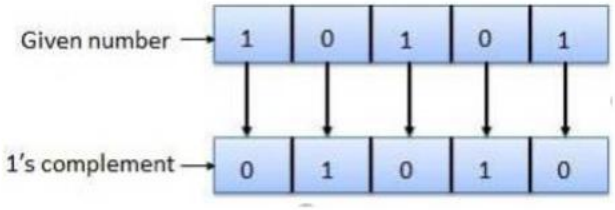
For binary number system (base 2), when we replace the value of r, we get

1. 2's Complement (r's complement)
2. 1's Complement ((r-1)'s complement)

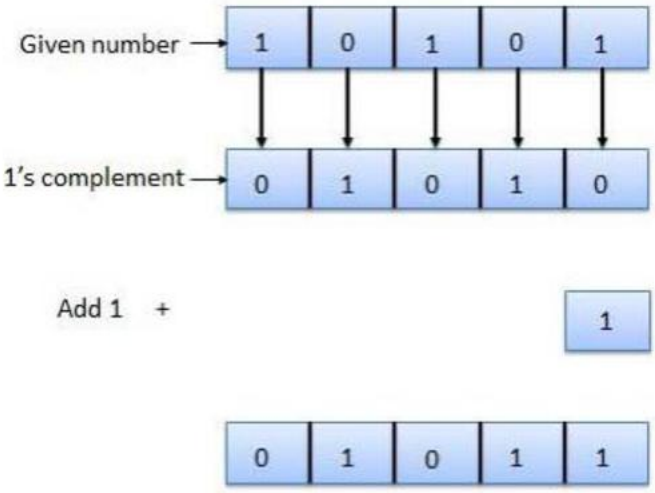
Similarly, for decimal Number system (base 10)

1. 10's complement
2. 9's complement

1's complement
The 1's complement of a number is found by changing all 1's to 0's and all 0's to 1's. This is called as taking complement or 1's complement. Example of 1's Complement is as follows.



2's complement
The 2's complement of binary number is obtained by adding 1 to the Least Significant Bit (LSB) of 1's complement of the number
2's complement = 1's complement + 1
Example of 2's Complement is as follows.



Subtraction using 1's Complement

The steps to subtract two binary number using 1's complement is explained as following below:

1. Make both numbers equal bits
2. Take 1's complement of the subtrahend
3. Add with minuend
4. If the result of above addition has carry bit 1, then add it to the least significant bit (LSB) of given result
5. If there is no carry bit 1, then take 1's complement of the result and prefix with –ve sign.

Example: Subtract 1110000 from 1100000

1`s complement of 1110000 =0001111
Adding it with minuend (i.e.1100000) =0001111
+1100000
1101111

Since, there exists` no any additional bits (carry),
1`s complement of 1101111 =0010000
Hence, the difference will be -0010000.
Example: 111000 -110000
1`s complement of 110000 =001111
Adding it with 111000 =001111
+111000
1000111

Since, there exists` one additional bit,
Difference
=000111
+1
001000

Hence, Difference will be 001000.

Subtraction using 2's Complement

The operation is carried out by means of the following steps:

1. Make equal no. of bits
2. At first, 2's complement of the subtrahend is found.
3. Then it is added to the minuend.
4. If the final carry over of the sum is 1, it is dropped and the result is positive.
5. If there is no carry over, the two's complement of the sum will be the result and it is negative.

Example: Subtract 1110000 from 1100000

2`s complement of 1110000 =0001111
+1
0010000

Adding it with minuend (i.e.1100000) =0010000
+1100000
1110000

Since, there exists` no any additional bits (carry),
2`s complement of 1110000 =0001111
+1
0010000

Hence, the difference will be -0010000.
Example: 111000 -110000
2`s complement of 110000 =001111
+1
010000

Adding it with 111000 =010000
+111000
1001000

Since, there exists` one additional bit,
Difference =001000 (Neglect carry i.e. 1)
Hence, Difference will be 001000.

9's and 10's Complement

The 9's complement of a decimal number can be obtained by subtracting each digit of the number from 9. For example: nine complement of 4 is 5(9-4=5) and 268 is (999-268=731)

10's complement of a number can be obtained by adding 1 to the least significant bit of 9's complement of that number. For Example: 10's complement of 123= (999-123+1=877)

Subtraction of decimal Number using 9's Complement

1. Make both numbers equal bits
2. Take 9's complement of the subtrahend
3. Add with minuend
4. If the result of above addition has additional digit, then add it to the least significant bit (LSB) of given result
5. If there is no additional digit, then take 9's complement of the result and prefix with –ve sign.

Subtraction of decimal Number using 10's Complement

1. Make both numbers equal bits

2. Take 10's complement of the subtrahend
3. Add with minuend
4. If the result of above addition has additional digit, then ignore it. Remaining bit is the answer.
5. If there is no additional digit, then take 10's complement of the result and prefix with –ve sign.

Addition and subtraction of binary Numbers

Binary Addition

Case	A+B	Carry	Sum
1	0 + 0	0	0
2	0 + 1	0	1
3	1 + 0	0	1
4	1 + 1	1	0

Binary Subtraction

Case	A-B	borrow	Subtract
1	0 - 0	0	0
2	1 - 0	0	1
3	1 - 1	0	0
4	0 - 1	1	0

Binary Codes

A binary code represents text, computer processor instructions, or any other data using a two-symbol system. The two-symbol system used is often "0" and "1" from the binary number system. The binary code assigns a pattern of binary digits, also known as bits, to each character, instruction, etc. It is possible to arrange n bit in 2^ndistinct ways. For example, a binary string of eight bits can represent any of (2^8=256) possible values and can, therefore, represent a wide variety of different items.

In the coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code. The digital data is represented, stored and transmitted as group of binary bits. This group is also called as **binary code**. The binary code is represented by the number as well as alphanumeric letter.

Advantages of Binary Code

- Binary codes are suitable for the computer applications.
- Binary codes are suitable for the digital communications.
- Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- Since only 0 & 1 are being used, implementation becomes easy.

BCD (Binary Coded Decimal)

BCD is a straight assignment of binary equivalent. Binary codes for decimal digits require a minimum of four bits.

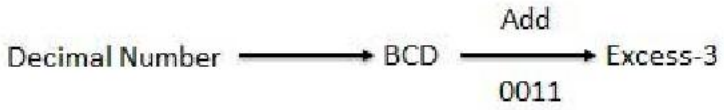
DecimalDigit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Numbers in digital computers are represented in binary or in decimal through a binary codes. When specifying data, the user likes to give the data in decimal form. The decimal numbers are converted to binary when arithmetic operations are done. It is very important to understand the difference between conversion of a decimal number to binary and the binary coding of a decimal Number.

- The bits obtained from conversion are binary digits
- Bits obtained from coding with combination of 0's and 1's according to the rule of code are binary coding.
- For example:
- When the decimal number 395 is converted into binary: 110001011
- The BCD representation of 395 is equal to: 0011 1001 0101

Excess-3 code

The Excess-3 code is also called as XS-3 code. It is non-weighted code used to express decimal numbers. The Excess-3 code words are derived from the BCD code words adding $(0011)_2$ or $(3)_{10}$ to each code word in BCD. The excess-3 codes are obtained as follows:



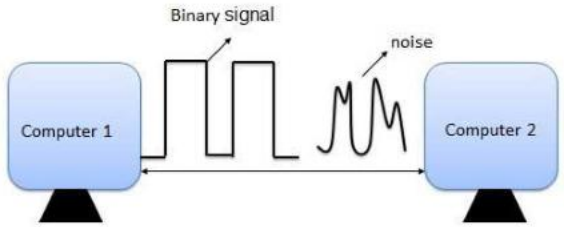
Example:

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Error detection codes

What is Error?

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.

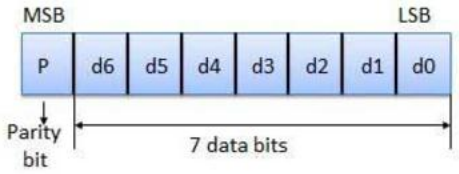


Whenever a message is transmitted, it may get twisted by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is **parity check.**

How to Detect and Correct Errors?

To detect and correct the errors, additional bits are added to the data bits at the time of transmission.
The additional bits are called **parity bits.** They allow detection or correction of the errors.

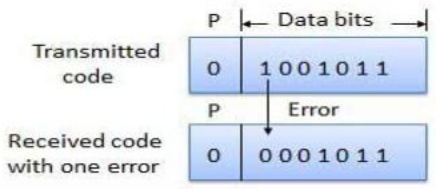
It is the simplest technique for detecting and correcting errors. The MSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.



Even parity -- Even parity means the number of 1's in the given word including the parity bit should be even (2,4,6,....). This bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is even.
Odd parity -- Odd parity means the number of 1's in the given word including the parity bit should be odd (1,3,5,....). This bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is odd.

How Does Error Detection Take Place?

Parity checking at the receiver can detect the presence of an error if the parity of the receiver signal is different from the expected parity. That means, if it is known that the parity of the transmitted signal is always going to be "even" and if the received signal has an odd parity, then the receiver can conclude that the received signal is not correct. If an error is detected, then the receiver will ignore the received byte and request for retransmission of the same byte to the transmitter.



Parity Bit Generator

There are two types of parity bit generators based on the type of parity bit being generated. Even parity generator generates an even parity bit. Similarly, odd parity generator generates an odd parity bit.

Even Parity Generator

Now, let us implement an even parity generator for a 3-bit binary input, WXY. It generates an even parity bit, P. If odd number of ones present in the input, then even parity bit, P should be '1' so that the resultant word contains even number of ones. For other combinations of input, even parity bit, P should be '0'. The following table shows the Truth table of even parity generator.

Binary Input WXY	Even Parity bit P
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

From the above Truth table, we can write the Boolean function for even parity bit as

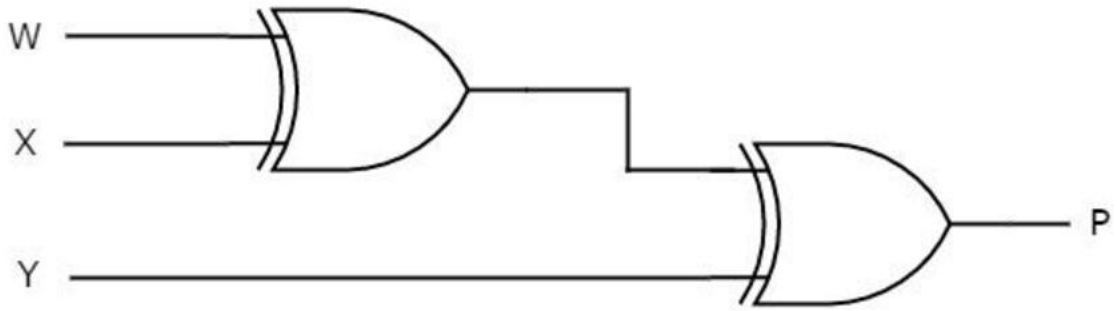
$P=W'X'Y+W'XY'+WX'Y'+WXY$

$\Rightarrow P=W'(X'Y+XY')+W(X'Y'+XY)$

$\Rightarrow P=W'(X \oplus Y)+W(X \oplus Y)'=W \oplus X \oplus Y$

* XNOR gate is the complement of XOR gate

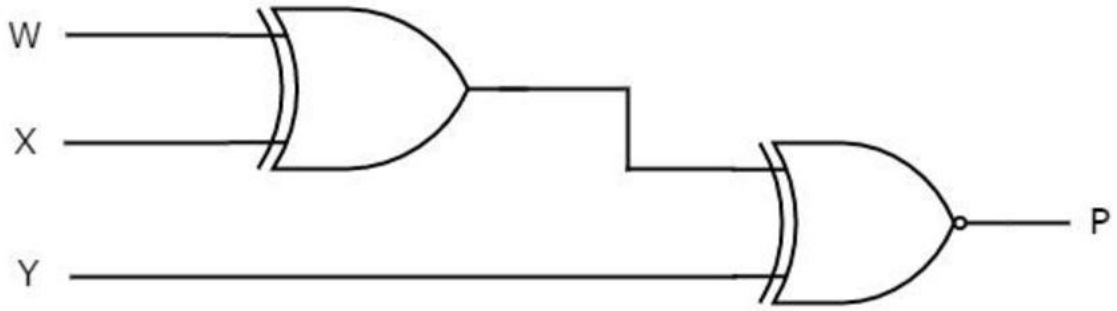
The following figure shows the circuit diagram of even parity generator.



Odd Parity Generator

If even number of one's present in the input, then odd parity bit, P should be '1' so that the resultant word contains odd number of ones. For other combinations of input, odd parity bit, P should be '0'.

Follow the same procedure of even parity generator for implementing odd parity generator. The circuit diagram of odd parity generator is shown in the following figure.



The above circuit diagram consists of Ex-OR gate in first level and Ex-NOR gate in second level. Since the odd parity is just opposite to even parity, we can place an inverter at the output of even parity generator. In that case, the first and second levels contain an ExOR gate in each level and third level consist of an inverter.

Assignment :

1. What is number system? Explain the types of number system.
2. Write the steps for subtracting the binary numbers using 1st and 2nd complement.
3. What is binary code? Write its advantages?
4. What is error? How to detect and correct errors ?
5. Convert (573)10=(?)2
6. Convert (58.515)10=(?)2
7. Convert (7893)10=(?)16
8. Convert (4989.612)10=(?)16
9. Convert (7326)10=(?)8
10. Convert (101111)2=(?)10
11. Convert (111011.1101)2=(?)10
12. Convert (7305)8=(?)10
13. Convert (7635.46)8=(?)10
14. Convert (7D9A)16=(?)10
15. Convert (9EDF.EA)16=(?)8

---

16. Convert (1011010)2=(?)8
17. Convert (1100.1101)2=(?)8
18. Convert (1011011)2=(?)16
19. Convert (11010.110)2=(?)16

---

20. Convert (7356)8=(?)2
21. Convert (625.7)8=(?)2

---

22. Convert (BC2A)16=(?)2
23. Convert (A5.7D)16=(?)2

---

24. Convert (735064)8=(?)16
25. Convert (8ABC)16=(?)8
26. Add (110011)2 and (100101)2
27. Add (111)2 and (100)2
28. Add (1111)2 and $(1000)_2$

---

29. Subtract (0101)2 from (1010)2
30. Subtract (011)2 from (1000)2
31. Subtract 1010 from 1100
32. Subtract 1111 from 11100
33. Subtract 11100 from 10101
34. Subtract 110 from 1110
35. Subtract 1011 from 100

36. Convert the decimal number 256 into base 2 number system.
37. Convert the binary numbers into decimal
    a. 10101
    b. 1101
    c. 10010
38. Obtain the first and second complement of the given numbers.
    a. 1010101
    b. 0111000
    c. 100000
39. Find the 9's and 10's complement of following decimal numbers
    a. 1653
    b. 2564
    c. 452

- End of Unit 1 -