Quadrant II- Notes

Paper Code: ELC 102

Module Name: Representation of signed and unsigned numbers, BCD code

Module Number:11

Representation of signed and unsigned numbers, BCD code

In decimal number system a plus (+) sign is used to represent a positive number and minus(-) sign to denote a negative number. The plus sign is usually dropped, and absence of any sign means the number has a positive value. This is the representation of signed numbers.

Digital systems can understand only two symbols 0 and 1 and therefore the same symbols must be used to indicate the sign of the number also.

Three representations in which signed binary numbers can be represented.

Sign Magnitude representation

One's complement representation.

Two's complement representation.

Sign-magnitude representation

For n-bit number the leftmost bit is reserved as a sign bit while the remaining (n-1) bits represent the magnitude. A +ve sign is represented by 0 bit whereas -ve sign is shown by 1 bit

10001 ⇔-1

Represent the following decimal numbers in sign-magnitude representation

1) +2 010
2) -35 1 100011
3) +100 0 1100100

Solution:

1) +2

   2/2 1 0
   1/2 0 1

   $2_{10} \Leftrightarrow 10_2$

   ∴ + 2 = 0 10

2) -35

We first find the binary equivalent of 35 as follows:

	Q	R
35/2	17	1
17/2	8	1
8/2	4	0
4/2	2	0
2/2	1	0
1/2	0	1

$\therefore 35_{10} = 100011_2$

$\therefore$ - 35 = 1 100011

A number system with base b or radix r has two complements.

radix's complement

(radix – 1)'s complement

For binary no. system radix =2

2's complement and 1's complement

1's complement of a binary number is obtained by subtracting each bit of the given binary no. from 1

Find 1's complement of 110

```
   1   1   1
 -   -   -
   1   1   0
 ---------------
   0   0   1
```

1's complement of 110 is 001

1's complement of 1110001010 is 0001110101

Thus, 1's complement of a binary number can be obtained by replacing each 1 by 0 and 0 by 1. Both numbers are complement of each other. If one of these number is positive, then the other number will be negative with the same magnitude and vice-versa. The one's complement method is widely used to represent signed binary numbers.

2's complement of a binary number

2's complement = 1's complement +1

Find 2's complement of 1101

= 1's complement +1

=0 0 1 0

+ 1

0 0 1 1

2’s complement of 1101 is 0011

Find the two’s complement of the following binary numbers:

1) 10101
2) 100
3) 1000

Solution:

1) 10101

   2’s complement =1’s complement +1

   = 01010

   + 1

   01011

2) 2’s complement of 100 = 1’s complement of 100 +1

   = 011

   $+_{1\ 1}$ 1

   = 100

3) 2’s complement of 1000 is = 0111+1 =1000

1’s complement representation:

A positive number is represented in the same way as in the sign magnitude representation.

To represent a negative number, we first find the binary equivalent of the given no. with positive sign and then find the 1’s complement of the resulting binary no.

Represent the following decimal numbers in 1’s complement representation.

1) + 4

	Q	R
4/2	2	0
2/2	1	0

$\therefore 4_{10} = 100_2$

$\therefore$ + 4 = 0 100

2) -25

$25_{10} = 11001_2$

+25 = 0 11001

1's complement of 011001 = 100110

∴-25 = 100110

Represent the following numbers in 1's complement representation.

+34 0 100010

-87

+87 0 1010111

∴-87 = 1's complement of 01010111 = 10101000

∴-87 = 10101000

2's complement representation:

A positive number is represented in the same way as in the sign magnitude representation.

To represent a negative number, we first find the binary equivalent of the given no. with positive sign and then find the 2's complement of the resulting binary no.

Represent the given numbers in 2's complement representation.

1) +4 = 0 100

2) -25

+ 25 = 0 11001

-25 = 2's complement of 011001= 100110 + 1

-25= 100111

	Sign magnitude	1's complement	2's complement
+25	011001	011001	011001
-25	111001	100110	100111

Binary Coded Decimal (BCD)

In BCD, a digit is usually represented by four bits which, in general, represent the values/digits/characters 0-9. Other bit combinations are sometimes used for sign or other indications. The most obvious way of encoding digits is *Natural BCD* , where each decimal digit is represented by its corresponding four-bit binary value, as shown in the following table. This is also called "8421" encoding.

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

This scheme can also be referred to as *Simple Binary-Coded Decimal* (*SBCD*) or *BCD 8421*, and is the most common encoding. Others include the so-called "4221" and "7421" encoding – named after the weighting used for the bits and excess-3 . For example, the BCD digit 6, `0110'b` in 8421 notation, is 11001_2 in 4221 (two encodings are possible), 0110_2 in 7421, while in Excess-3 it is 1001_2

10 in BCD⇔ 0001 0000 10 in binary⇔ 1010

25 in BCD ⇔ 0010 0101 BINARY ⇔ 11001

0110 0101 0011 ⇔653

011001010011⇔ 1+2+16+64+512+ 1024 = 1619

BCD is a weighted code It is 4 bit code.

It takes more number of bits to code a decimal number in BCD code than using straight binary code. However, inspite of this disadvantage, it is convenient and useful code for input and output operations in digital systems.

BCD is very common in electronic systems where a numeric value is to be displayed, especially in systems consisting solely of digital logic, and not containing a microprocessor. By employing BCD, the manipulation of numerical data for display can be greatly simplified by treating each digit as a separate single sub-circuit. This matches much more closely the physical reality of display hardware—a designer might choose to use a series of separate identical seven-segment displays to build a metering circuit, for example. If the numeric quantity were stored and manipulated as pure binary, interfacing with such a display would require complex circuitry. Therefore, in cases where the calculations are

relatively simple, working throughout with BCD can lead to an overall simpler system than converting to and from binary. Most pocket calculators do all their calculations in BCD.

The same argument applies when hardware of this type uses an embedded microcontroller or other small processor. Often, representing numbers internally in BCD format results in smaller code, since a conversion from or to binary representation can be expensive on such limited processors. For these applications, some small processors feature dedicated arithmetic modes, which assist when writing routines that manipulate BCD quantities.

As most computers deal with data in 8-bit bytes it is possible to use one of the following methods to encode a BCD number:

- **Unpacked**: Each decimal digit is encoded into one byte, with four bits representing the number and the remaining bits having no significance.
- **Packed**: Two decimal digits are encoded into a single byte, with one digit in the least significant nibble(bits 0 through 3) and the other numeral in the most significant nibble (bits 4 through 7).

As an example, encoding the decimal number **91** using unpacked BCD results in the following binary pattern of two bytes:

Decimal: 9 1

Binary : 0000 1001 0000 0001

In packed BCD, the same number would fit into a single byte:

Decimal: 9 1

Binary: 1001 0001

Advantages of BCD Codes

- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.
- It is easy to encode and decode decimals into BCD and vice versa.
- It is also simple to implement a hardware algorithm for the BCD converter.
- It is very useful in digital systems whenever decimal information is given either as inputs or displayed as outputs.
- Digital voltmeters, frequency converters and digital clocks all use BCD as they display output information in decimal

Disadvantages of BCD Codes

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.

- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

• BCD code for a given decimal number requires more bits than the straight binary code and hence there is difficulty to represent the BCD form in high speed digital computers in arithmetic operations, especially when the size and capacity of their internal registers are restricted or limited.

• The arithmetic operations using BCD code require a complex design of Arithmetic and 'logic Unit (ALU) than the straight "binary number system.

• The speed of the arithmetic operations that can be realized using BCD code is naturally slow due to the complex hardware circuitry involved.