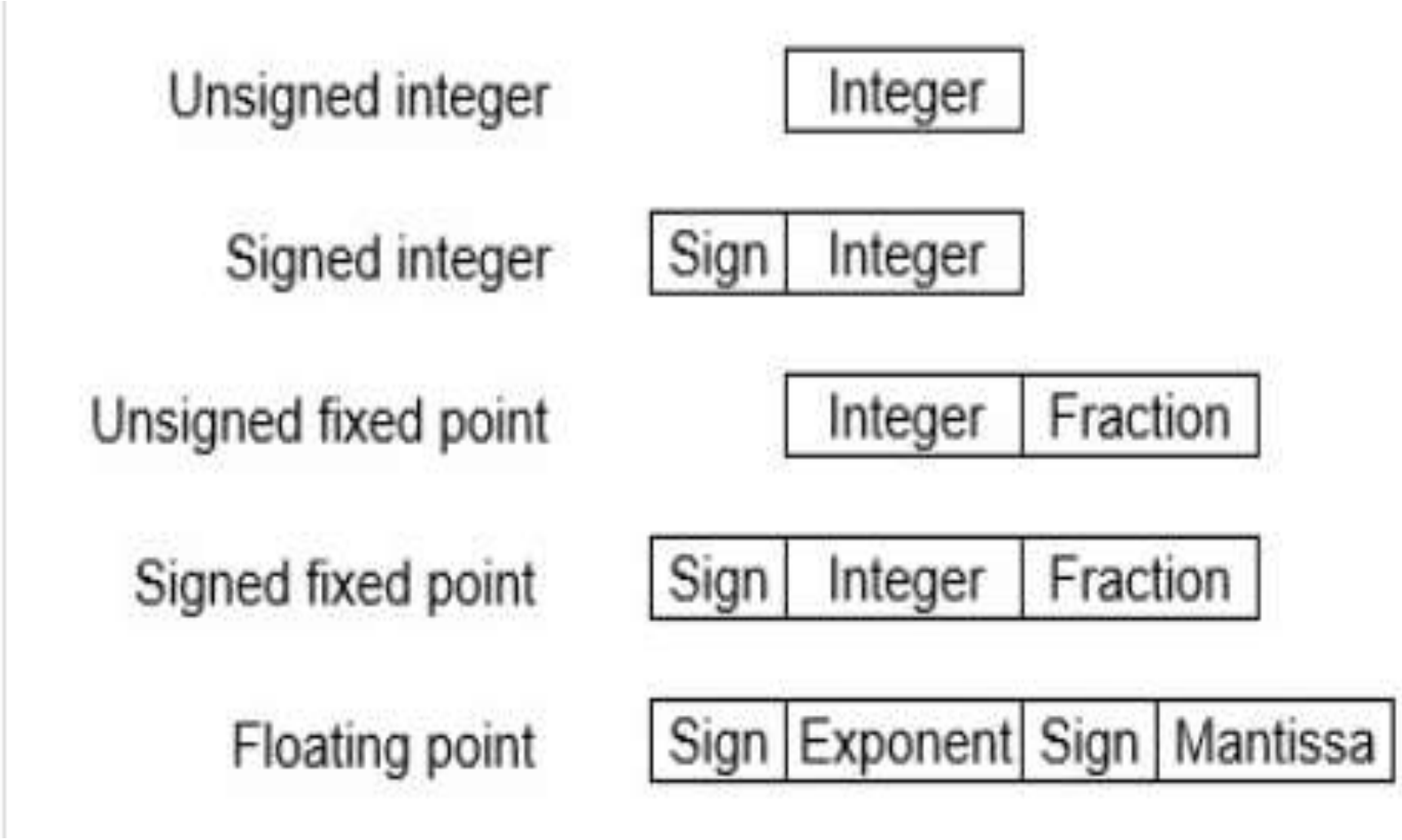
Lecture 9

Dr. Krishan Arora
Asstt. Prof. and Head

Storing Real Number



- There are two major approaches to store real numbers (i.e., numbers with fractional component) in modern computing. These are
- (i) Fixed Point Notation and
- (ii) Floating Point Notation.
- In fixed point notation, there are a fixed number of digits after the decimal point, whereas floating point number allows for a varying number of digits after the decimal point.

Fixed-Point Representation

- This representation has fixed number of bits for integer part and for fractional part. For example, if given fixed-point representation is IIII.FFFF, then you can store minimum value is 0000.0001 and maximum value is 9999.9999. There are three parts of a fixed-point number representation: the sign field, integer field, and fractional field.

Example

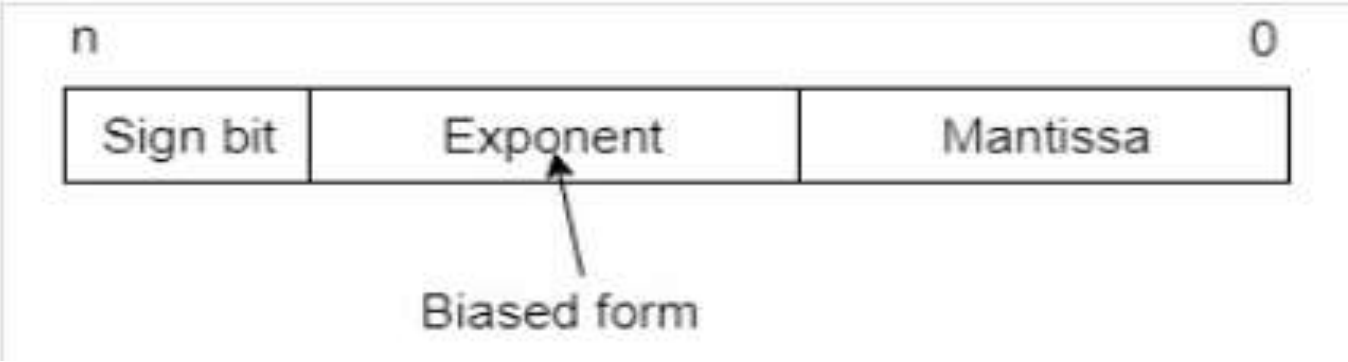
Assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.

Then, -43.625 is represented as following:

- Where, 0 is used to represent + and 1 is used to represent -. 000000000101011 is 15 bit binary value for decimal 43 and 1010000000000000 is 16 bit binary value for fractional 0.625.

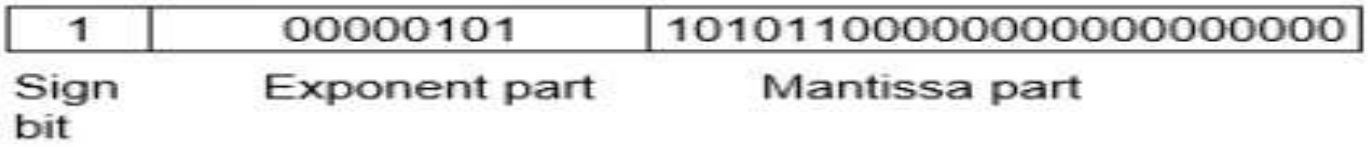
Floating-Point Representation

- The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent.



Example

- Suppose number is using 32-bit format: the 1 bit sign bit, 8 bits for signed exponent, and 23 bits for the fractional part.
- Then −53.5 is normalized as $-53.5=(-110101.1)_2=$ $(-1.101011)x2^5$, which is represented as following below,



- Where 00000101 is the 8-bit binary value of exponent value +5.

Practice Questions

- Find EX-3 code of Decimal number 81.61

Explanation

Decimal	BCD	Excess-3
8	1000+0011	1011
1	0001+0011	0100
6	0110+0011	1001

3) So, the excess-3 code of the decimal number 81.61 is **1011 0100.1001 0100**

Excess 3 Code Addition

- The operation of addition can be done by very simple method we will illustrate the operation in a simple way using steps.
- Step 1
  We have to convert the numbers (which are to be added) into excess 3 forms by adding 0011 with each of the four bit groups them or simply increasing them by 3.
  Step 2
  Now the two numbers are added using the basic laws of binary addition, there is no exception for this method.
  Step 3
  Now which of the four groups have produced a carry we have to add 0011 with them and subtract 0011 from the groups which have not produced a carry during the addition.
  Step 4
  The result which we have obtained after this operation is in Excess 3 form and this is our desired result

Example

- 0011 0101 0110 and 0101 0111 1001 are the two binary numbers.
- Now following the first step we take the excess 3 form of these two numbers which are 0110 1000 1001 and 1000 1010 1100, now these numbers are added following the basic rules of addition.

```
  0110 1000 1001
  1000 1010 1100
-------------------
  1111 0011 0101
```

- Now adding 0011 to the groups which produces a carry and subtracting 0011 from the groups which did not produced carry we get the result as 1100 0110 1000

Excess 3 Code Subtraction

- Step 1
  Like the previous method both the numbers have to be converted into excess 3 code
- Step 2
  Following the basic methods of binary subtraction, subtraction is done.
  Step 3
  Subtract ‘0011’ from each BCD four-bit group in the answer if the subtraction operation of the relevant four-bit groups required a borrow from the next higher adjacent four-bit group.
  Step 4
  Add ‘0011’ to the remaining four-bit groups, if any, in the result.
  Step 5
  Finally we get the desired result in **excess 3 code**.

Example

Subtract 0001 1000 0101 and 0000 0000 1000 using Excess-3

Step 1

Now the excess 3 equivalent of those numbers are 0100 1011 1000 and 0011 0011 1011

Now performing the operation of binary subtraction we get

$$\begin{array}{r} 0100\ 1011\ 1000 \\ 0011\ 0011\ 1011 \\ \hline 0001\ 0111\ 1101 \end{array}$$

- Now in the above mentioned operation the least significant column which needed a borrow and the other two columns did not need borrow. Now we have to subtract 0011 from the result of this column and add 0011 to the other two columns, we get 0100 1010 1010. This is the result expressed in **excess 3 codes**.

Quick Quiz (Poll 1)

- The excess-3 code for 597 is given by _____

  a) 100011001010
  b) 100010100111
  c) 010110010111
  d) 010110101101

Solution

- Answer: a
  Explanation: The addition of ‘3’ to each digit yields the three new digits ‘8’, ’12’ and ’10’. Hence, the corresponding four-bit binary equivalents are 100011001010, in accordance to 8421 format.

Quick Quiz (Poll 2)

- The decimal equivalent of the excess-3 number 110010100011.01110101 is

  _____

  a) 970.42
  b) 1253.75
  c) 861.75
  d) 1132.87

Solution

- Answer: a
  Explanation: The conversion of binary numbers into digits ‘1100’, ‘1010’, ‘0011’, ‘0111’ and ‘0101’ gives ’12’, ‘10’, ‘3’, ‘7’ and ‘5’ respectively. Hence, the decimal number is 970.42.