Module 2

Commonly used number systems. Fixed and floating-point representation of numbers;

number systems

The number system or the numeral system is the system of naming or representing numbers.

Types of Number System:

- Decimal number system (Base- 10)
- Binary number system (Base- 2)
- Octal number system (Base-8)
- Hexadecimal number system (Base- 16)

number systems

Decimal Number System (Base 10 Number System)

The decimal number system has a base 10 because it uses ten digits from 0 to 9

The decimal number 1457 consists of the digit 7 in the units position, 5 in the tens place, 4 in the hundreds position, and 1 in the thousands place whose value can be written as

$(1\times10^3) + (4\times10^2) + (5\times10^1) + (7\times10^0)$

$=(1\times1000) + (4\times100) + (5\times10) + (7\times1)$

$=1000 + 400 + 50 + 7$

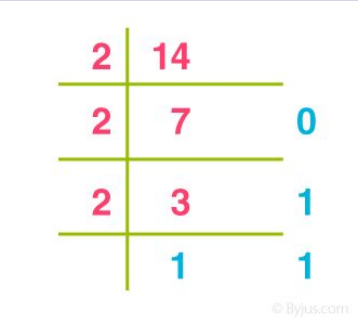
$=1457$

number systems

Binary Number System (Base 2 Number System)

The base 2 number system is also known the Binary number system wherein, only two binary digits exist, i.e., 0 and 1. Specifically, the usual base-2 is a radix of 2

Write $(14)_{10}$ as a binary number.



number systems

Octal Number System (Base 8 Number System)

- **Example: Convert 215_8 into decimal.**
- **Solution:**
- $215_8 = 2 \times 8^2 + 1 \times 8^1 + 5 \times 8^0$
- $= 2 \times 64 + 1 \times 8 + 5 \times 1$
- $= 128 + 8 + 5$
- $= 141_{10}$

number systems

Hexadecimal Number System (Base 16 Number System)

In the hexadecimal system, numbers are written or represented with base 16. In the hex system, the numbers are first represented just like in decimal system, i.e. from 0 to 9. Then, the numbers are represented using the alphabets from A to F.

Hexadecimal	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Decimal	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**

number systems

- **Example 1:**
- Convert $(1056)_{16}$ to octal number.
- **Solution:**
- Given, 1056_{16} is an hex number.
- First we need to convert the given hexadecimal number into decimal number
- $(1056)_{16}$
- $= 1 \times 16^3 + 0 \times 16^2 + 5 \times 16^1 + 6 \times 160$
- $= 4096 + 0 + 80 + 6$
- $= (4182)_{10}$

- Now, we have to convert 4182_{10} to octal
- 4182 / 8 = 522 with remainder 6
  522 / 8 = 65 with remainder 2
  65 / 8 = 8 with remainder 1
  8 / 8 = 1 with remainder 0
  1 / 8 = 0 with remainder 1

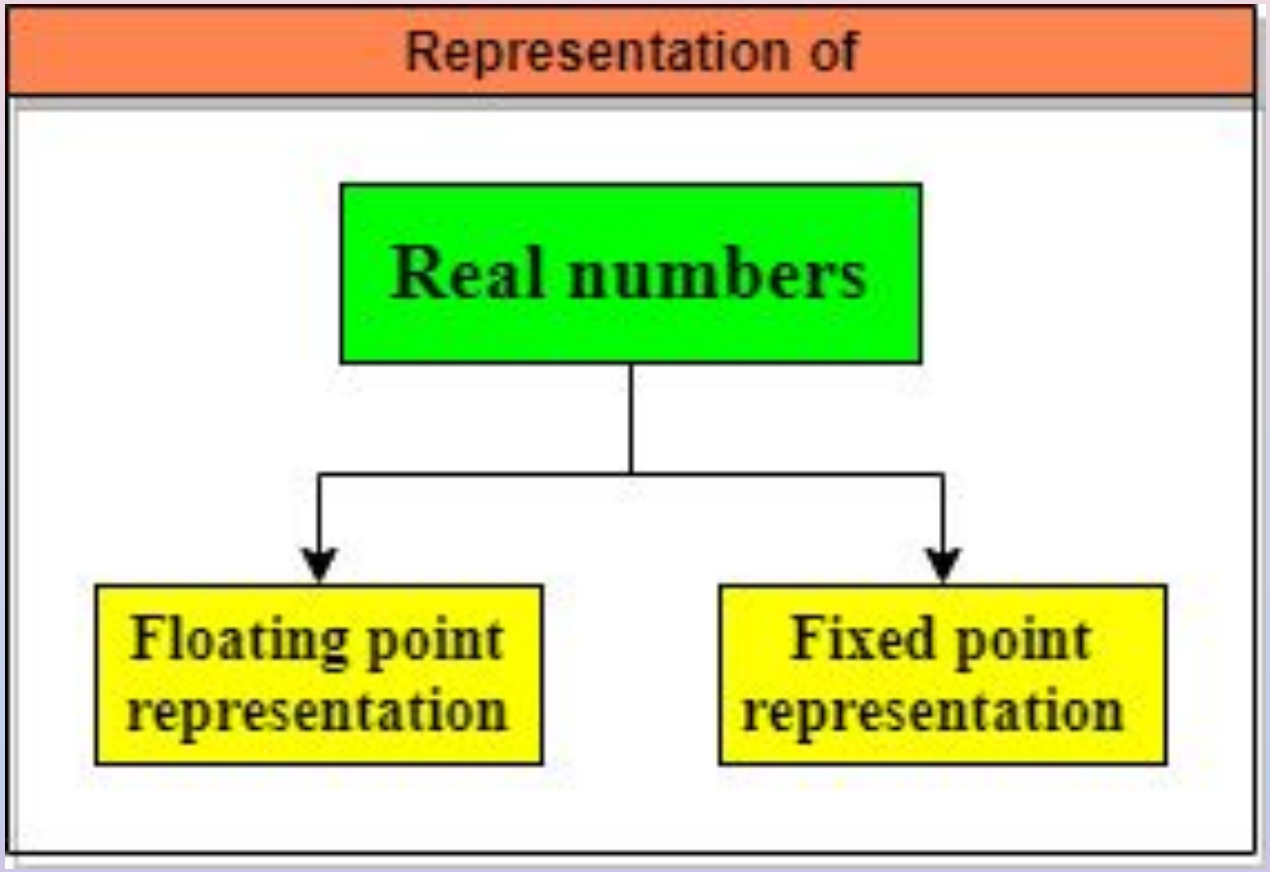
- Then just write down the remainders in the reverse order to get the answer, The hexadecimal number 1056 converted to octal is therefore equal to :

  10126

number systems

- **Example 2:**
- Convert hexadecimal 2C to decimal number.
- **Solution:**
- Step by step solution
- **Step 1**: Write down the hexadecimal number:
- $(2C)_{16}$
- **Step 2**: Show each digit place as an increasing power of 16:
- $2x16^1 + Cx16^0$
- **Step 3**: Convert each hexadecimal digits values to decimal values then perform the math:
- $2x16 + 12x1 = (44)_{10}$
- So, the number 44 is the decimal equivalent of hexadecimal number 2C (Answer).

Fixed and floating-point representation



Fixed and floating-point representation

Fixed point representation

In computing, fixed-point number representation is a real data type for a number. With the help of fixed number representation, data is converted into binary form, and then data is processed, stored and used by the system.

Fixed point representation of data

Sign bit -The fixed-point numbers in binary uses a sign bit. A positive number has a sign bit 0, while a negative number has a sign bit 1.

Integral Part – The integral part is of different lengths at different places. It depends on the register's size, like in an 8-bit register, integral part is 4 bits.

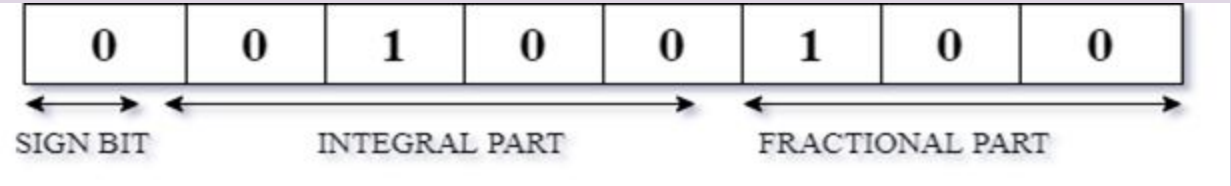
- **Fractional part** – Fractional part is also of different lengths at different places. It depends on the register's size, like in an 8-bit register, integral part is of 3 bits.
- 8 bits = 1Sign bit + 4 bits(integral) + 3bits (fractional part)
- 16 bits = 1Sign bit + 9 bits(integral) +6 bits (fractional part)
- 32 bits = 1Sign bit + 15 bits(integral) + 9 bits (fractional part)

How to write the number in Fixed-point notation?

Number is 4.5

Step 1:- Convert the number into binary form. 4.5 = 100.1

Step 2:- Represent binary number in Fixed point notation



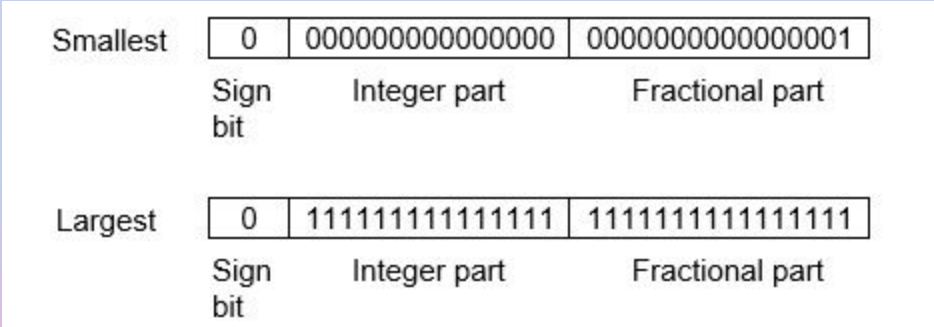
Question: Assume a computer in which a number is stored using fixed point notation (16 bits):which reserve 7 bits for the integer part and 9 bits for fractional part. Represent 43.625 ??

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

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.

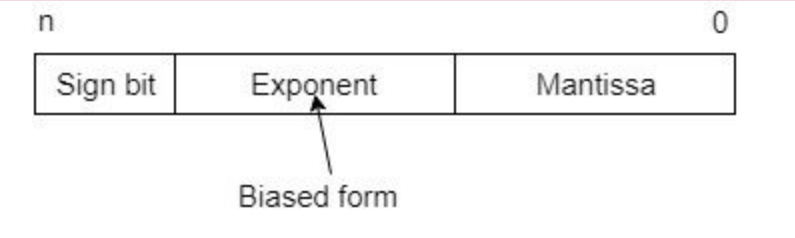
- Then, -43.625 is represented as

- 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.
- The advantage of using a fixed-point representation is performance and disadvantage is relatively limited range of values that they can represent. So, it is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy. A number whose representation exceeds 32 bits would have to be stored inexactly.



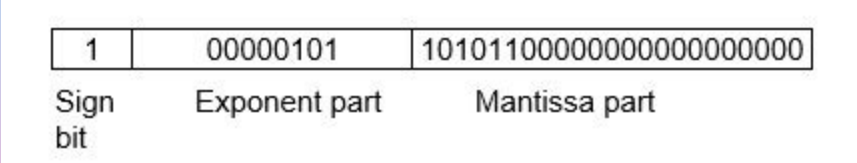
- These are above smallest positive number and largest positive number which can be store in 32-bit representation as given above format. Therefore, the smallest positive number is $2^{-16} \approx 0.000015$ approximate and the largest positive number is $(2^{15}-1)+(1-2^{-16})=2^{15}(1-2^{-16})$ $=32768$, and gap between these numbers is 2^{-16}.
- We can move the radix point either left or right with the help of only integer field is 1.

- **Floating-Point Representation –**
- This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).
- 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. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: Mxr^e.
- Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.

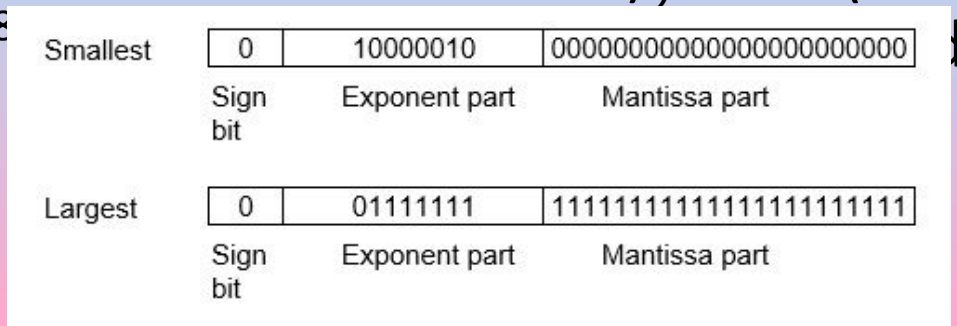


So, actual number is $(-1)^s(1+m)x2^{(e-Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.
Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.
The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $\pm(1.b_1b_2b_3 ...)_2x2^n$ This is normalized form of a number x.

- **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. The leading bit 1 is not stored (as it is always 1 for a normalized number) and is referred to as a *"hidden bit*".
- Then −53.5 is normalized as $-53.5=(-110101.1)_2=(-1.101011)x2^5$, which is represented as following below,



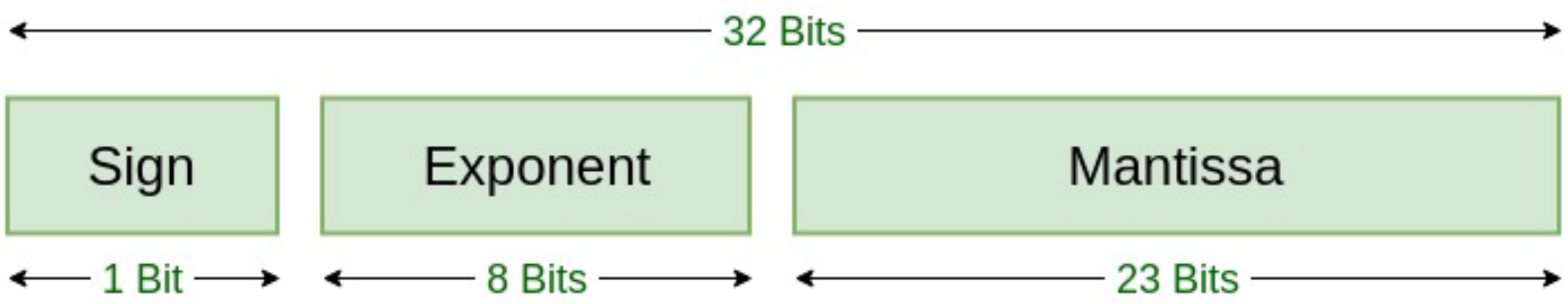
- Where 00000101 is the 8-bit binary value of exponent value +5.
- Note that 8-bit exponent field is used to store integer exponents $-126 \leq n \leq 127$.
- The smallest normalized positive number that fits into 32 bits is $(1.00000000000000000000000)_2 x 2^{-126} = 2^{-126} \approx 1.18 x 10^{-38}$, and largest normalized positive number that fits into 32 bits is $(1.11111111111111111111111)_2 x 2^{127} = (2^{24}-1) x 2^{104} \approx 3.40 x 10^{38}$ as following below,



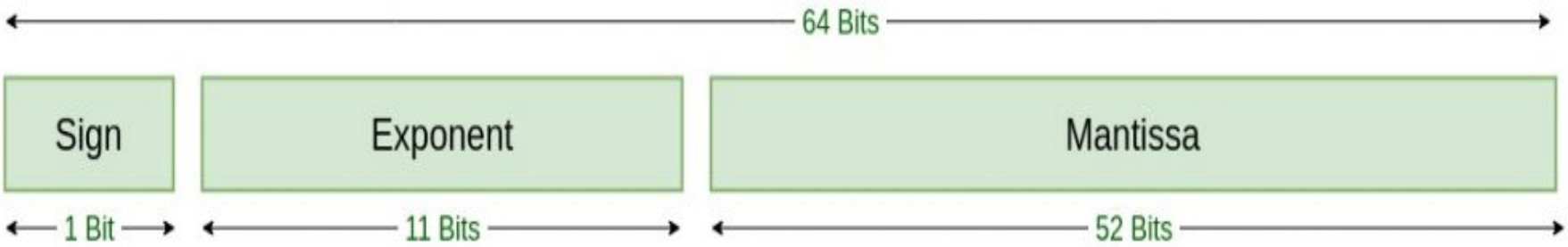
- The precision of a floating-point format is the number of positions reserved for binary digits plus one (for the hidden bit). In the examples considered here the precision is 23+1=24.
- The gap between 1 and the next normalized floating-point number is known as machine epsilon. the gap is $(1+2^{-23})-1=2^{-23}$for above example, but this is same as the smallest positive floating-point number because of non-uniform spacing unlike in the fixed-point scenario.
- Note that non-terminating binary numbers can be represented in floating point representation, e.g., 1/3 = $(0.010101\ ...)_2$ cannot be a floating-point number as its binary representation is non-terminating.

IEEE 754 standard

- Half Precision (16 bit): 1 sign bit, 5 bit exponent, and 10 bit mantissa
- Single Precision (32 bit): 1 sign bit, 8 bit exponent, and 23 bit mantissa
- Double Precision (64 bit): 1 sign bit, 11 bit exponent, and 52 bit mantissa
- Quadruple Precision (128 bit): 1 sign bit, 15 bit exponent, and 112 bit mantissa



Single Precision
IEEE 754 Floating-Point Standard



Double Precision
IEEE 754 Floating-Point Standard

YPES	SIGN	BIASED EXPONENT	NORMALISED MANTISA	BIAS
Single precision	1(31st bit)	8(30-23)	23(22-0)	127
Double precision	1(63rd bit)	11(62-52)	52(51-0)	1023

1. Represent $(32.75)_{10}$ in IEEE 754 single and double precision format.

2. Assume a computer in which a number is stored using fixed point notation (16 bits):which reserve 7 bits for the integer part and 9 bits for fractional part. Represent 43.625 ??

3. Convert 2158 into decimal.

4. Convert $(1056)_{16}$ to octal number

5. Convert hexadecimal 2C to decimal number